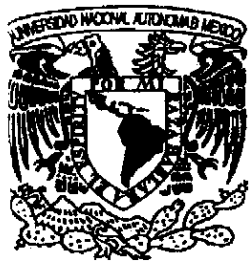

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DIVISION DE ESTUDIOS DE POSGRADO

"GENERACION DE NUMEROS PSEUDOALEATORIOS EFICIENTES EN MICROCOMPUTADORES"

TESIS

QUE PRESENTA

ANGEL ADOLFO OLMOS CERVANTES

PARA OBTENER EL GRADO DE:

DOCTOR EN INGENIERIA
(INVESTIGACION DE OPERACIONES)

DIRECTOR DE TESIS
DR. JOSE JESUS ACOSTA FLORES

MEXICO, D.F.

INDICE

1. INTRODUCCION

Un aspecto importante en el estudio de sistemas complejos, mediante modelos de simulación, lo constituye la generación de números aleatorios.

Las técnicas de Montecarlo, a pesar de haberse originado en 1876 con los trabajos de E. L. De Forest, recién pudieron aplicarse intensivamente con el advenimiento de las computadoras, en virtud de sus capacidades de cálculo en constante incremento, desde los inicios de la década de los sesenta, con diez mil operaciones aritméticas por segundo, hasta la década pasada, en que la cantidad de operaciones ascendía a miles de millones por segundo.

Las metodologías de simulación requieren, por una parte, disponer de una fuente, llamada generador de números aleatorios, de buena calidad estadística por los números pseudoaleatorios que produce; y por otra, tener acceso a generadores rápidos que reduzcan los costos insumidos en el tiempo de procesamiento.

Cuando las variables utilizadas en un proceso de simulación de un sistema determinado, no son estadísticamente independientes, los resultados que se infieren pueden estar afectados de importantes errores, debido al conocido fenómeno de correlación serial o autocorrelación de datos.

Desde los análisis de las propiedades de autocorrelación en 28 generadores elaborados por Coveyou y Macpherson (Coveyou R.R. y Macpherson R.D., 1960), se han realizado numerosos estudios sobre las condiciones que deben satisfacer las sucesiones de números pseudoaleatorios producidos por generadores, en particular, aquéllos que son congruenciales lineales.

Se considera que un generador es eficiente cuando proporciona sucesiones de números pseudoaleatorios que tengan una distribución uniforme y sean estadísticamente independientes; que puedan ser reproducibles por el mismo algoritmo; rápidos y de fácil programación, incluso para usarse con técnicas de paralelización; de buen funcionamiento y portabilidad; no repetitivas durante cualquier período dado, y que requieran una mínima cantidad de memoria del procesador.

En relación con lo antes expuesto, es frecuente verificar la carencia de sistemas que contengan herramientas destinadas a generar y probar sucesiones de números pseudoaleatorios, en particular en nuestras Universidades, para las actividades de

enseñanza e investigación, tales como modelado con técnicas Montecarlo, experimentos en Estadística, algoritmos probabilísticos, cálculo numérico, aplicaciones criptográficas, protocolos de seguridad en comunicaciones, etc.

Cuando en una simulación no se utilizan generadores de buena calidad, la transformación de sucesiones de números pseudoaleatorios en variables aleatorias, según una determinada distribución, pueden conducir a resultados sesgados o totalmente erróneos, como lo hemos mostrado experimentalmente en una aplicación real (cfr. 4.3.3.2 y 5.2.2).

En consecuencia, en nuestro trabajo de tesis nos proponemos alcanzar los siguientes objetivos:

a) Actualización y clasificación de generadores de números pseudoaleatorios (GNA), de acuerdo a la teoría y los algoritmos más recientes.

b) Incorporación de nuevas técnicas de programación de los GNA que sean compatibles con las arquitecturas de PC.

c) Verificación de la calidad estadística de los generadores con polinomios de Weyl.

d) Nuevo procedimiento para la selección de un GNA en base a una batería de pruebas empíricas.

e) Nuevos criterios de entropía para la evaluación de sucesiones de números pseudoaleatorios.

f) Nueva metodología para la elección de GNA eficientes.

g) Provisión de documentación suficientemente amplia en relación a los GNA y las Pruebas Teóricas y Empíricas de Aleatoriedad.

h) Nueva técnica para incorporar distintos GNA en una aplicación de simulación que requiera variables aleatorias simultáneas en un mismo proceso.

i) Desarrollo de un nuevo paquete de programas, denominado GENTSTPC, bajo los más usuales sistemas operativos de PC, con el propósito de generar, probar y almacenar archivos de números pseudoaleatorios, incluyendo la posibilidad de incorporar sucesiones producidas por nuevos generadores, con el objeto de verificar propiedades estadísticas de aleatoriedad.

Los lenguajes más empleados en los programas del sistema Gentstpc, son Fortran y C, además de otros paquetes matemáticos para cálculo numérico y simbólico, como Maple V, además de Matlab y Gauss para comprobar la eficiencia de sus propios generadores.

En cuanto a las Aplicaciones (Cap.5), se han utilizado Fortran, C, Java y GPSS/PC y GPSS World.

El Manual del Usuario, Anexo II, complementa la información que el programa de entrada de datos ofrece a través de numerosas ayudas.

La organización de este trabajo comprende 6 Capítulos y 2 Anexos:

1- Introducción

2- Estado del arte

3- Métodos de generación y pruebas estadísticas

4- Sistema Gentstpc

5- Aplicaciones

6- Conclusiones y recomendaciones

Anexos I y II

El **Cap.1** describe el marco científico e histórico de los números aleatorios y sus pruebas estadísticas; las condiciones que requieren los generadores de números pseudoaleatorios y los problemas de autocorrelación serial; las metas alcanzadas por el trabajo de tesis; y un resumen del contenido de sus Capítulos y Apéndices.

El **Cap.2** describe las ideas básicas de aleatoriedad y luego detalla el estado del arte relativo a otros tipos de Generadores y Pruebas; los proyectos e investigaciones en curso; y las nuevas aplicaciones de los números pseudoaleatorios, según la información más reciente sobre estos temas obtenida básicamente de Internet.

El **Cap.3** está dedicado a los generadores de sucesiones de números pseudoaleatorios y las pruebas de aleatoriedad. Comienza con una breve revisión de los más usuales métodos de congruencias lineales, y sus fundamentos teóricos, para luego introducir los algoritmos y pseudocódigos que forman el nuevo conjunto de generadores del paquete Gentstpc.

A continuación se construyen sucesiones de números pseudoaleatorios, generados por los polinomios de Weyl, empleando aproximaciones de números irracionales, para lo cual se ha elaborado un algoritmo original de cálculo.

La parte consagrada a las pruebas que verifican la calidad de los números pseudoaleatorios, abarca las pruebas de caracter empírico y del tipo teórico; y la siguiente, las pruebas de números aleatorios mediante entropía continua o discreta, a cuyo efecto se proporcionan algunos algoritmos sobre los avances teóricos más recientes en este tipo de pruebas.

El **Cap.4**, sistema GENTSTPC, es la descripción de los aportes originales a la tesis en cuanto a Generadores, Pruebas Empíricas, Criterios de Selección y una nueva metodología para determinar la eficiencia de un GNA, basada en pruebas de uniformidad y estimadores de entropía.

En el **Cap.5**, se sintetizan dos tipos de aplicaciones: las evaluaciones de los generadores en algunos lenguajes y paquetes estadístico-matemáticos disponibles en la Universidad Nacional del Comahue(UNC); y por otra parte, el empleo de generadores, que superan satisfactoriamente las pruebas de calidad estadística, en modelos de simulación. En este último caso, se incluyen tres aplicaciones: un estudio sobre grafos Pert; una investigación sobre modelos de precipitaciones de granizo en la región de influencia de la UNC; y otra, acerca del rendimiento de distintas aeronaves para combatir incendios forestales en la precordillera de los Andes Patagónicos.

El **Cap.6**, resume las conclusiones y propuestas que surgen del tra
tesis.

Finalmente, el **Anexo I**, contiene los antecedentes del GENTSTPC re
los GNA y las Pruebas de Aleatoriedad; y en el **Anexo II** (CD-ROM) se i
manual del usuario con ejemplos y recomendaciones y un código fuente c
del sistema GENTSTPC.

Teniendo en cuenta que en este trabajo convergen las disciplinas cie
Investigación Operativa, Informática y Estadística, las metodologías a
pueden describirse sucintamente en forma separada.

La metodología empleada en simulación consta de las etapas: formula
problema; recolección de datos; diseño de un modelo matemático y su val
elaboración de un modelo computacional y verificación; validación; diseñ
perimentos; ejecución de programas; análisis de los resultados; documen
reportes de los resultados.

La metodología estadística más general aplicada al proceso de constru
modelos, a partir de datos empíricos, sigue las etapas: diseño del expe
estructuración del modelo; estimación de parámetros; comparación de la
de ajuste; verificación del modelo y simulación.

Los resultados esperados con las aportaciones de este trabajo serán
ciosas, en particular, para las Universidades, por sus aportes académicos
la región de influencia de la Universidad Nacional del Comahue, por sus c
ciones para la lucha contra las adversidades climáticas; la protección de
frutihortícolas, ante tormentas de granizo; los recursos forestales, ante lo
dios; la producción; el trabajo y el medio ambiente.

RESUMEN

En la actualidad, se reconoce a la tarea de generación de números pseudoaleatorios como un aspecto crucial en el empleo de metodologías modernas para la simulación de modelos. Cuando las variables utilizadas en un proceso de simulación de un sistema determinado no son estadísticamente independientes, los resultados que se infieren pueden adolecer de importantes errores, debido al conocido fenómeno de autocorrelación de datos.

En el presente trabajo de tesis, *Generación de números pseudoaleatorios en microcomputadores*, se investiga por una parte, un nuevo conjunto de métodos analíticos para reproducir sucesiones de números, que posean determinadas propiedades de aleatoriedad, por lo que se designan pseudoaleatorios; que tengan períodos suficientemente extensos; y que puedan ser programables en arquitecturas compatibles con los microcomputadores del tipo PC.

Por otra parte, para evaluar la calidad de las sucesiones de números pseudoaleatorios, se realizan contrastes entre las baterías de pruebas estadísticas (teóricas y empíricas) existentes, y se introducen pruebas, de acuerdo a los más recientes enfoques para estudiar uniformidad e independencia, según el concepto de entropía, desde la teoría de las Probabilidades y la Estadística.

El análisis de tales sucesiones, que abarca generadores del tipo congruencial lineal y otros basados en polinomios de Weyl, se extiende a los generadores de números aleatorios (GNA) provistos por los fabricantes, en algunos paquetes estadísticos-matemáticos de uso frecuente, con el fin de detectar los generadores ineficientes causantes de fenómenos de correlación cuando se los utiliza en aplicaciones de gran porte.

Se muestra además, el comportamiento de varios GNA, cuando se incorporan simultáneamente a un mismo proceso de simulación de un sistema, corroborando de esa manera los resultados satisfactorios de una nueva metodología basada en un conjunto de pruebas empíricas y teóricas para la selección de generadores eficientes.

Con el fin de proveer la carencia de sistemas que contengan herramientas destinadas a generar y probar sucesiones de números pseudoaleatorios, especialmente en nuestras Universidades, para actividades de enseñanza e investigación,

ABSTRACT

Currently, the task of generating pseudo-random numbers is recognized as a crucial aspect in the use of modern methodologies for model simulation. Indeed, when the random variates used in the simulation process of a given system are not statistically independent, the results inferred may have important errors, due to the phenomenon known as data serial correlation.

At the present thesis work, *Generation of efficient pseudo-random numbers in microcomputers*, is investigated, on one hand, a new group of analytic methods to reproduce number sequences, which may have specific aleatory qualities; as a consequence they are called pseudo-random; they may also have sufficient extensive periods; moreover they may be programmable in compatible architectures with PC microcomputers type.

On the other hand, in order to evaluate the pseudo-random numbers sequences quality, are created contrasts between the existent statistics test batteries (theoretical and empirical), and are introduced tests, based on the most recent approach to study uniformity and independence, according to the concept of entropy, from the probabilities theory and statistics.

The analysis of such sequences, which embraces generators of the lineal congruential type and others based on Weyl's polynomials, is extended to the random numbers generators (RNG) provided by manufacturers in some frequently used statistics-mathematical packages, in order to detecting the inefficient generators that cause correlation phenomena when used in big applications.

The behavior of various RNG is also shown when they are simultaneously incorporated to a same simulation process of a system, corroborating by this way the satisfactory results of a new methodology based on a group of empirical and theoretical tests for selecting efficient generators.

Willing to provide the lack of systems which contains tools destined to generate and test pseudo-random numbers sequences, especially in our Universities, for teaching and investigating activities, the new contributions are combined in the development of a new package of programs, named GENTSTPC to be executed in PC microprocessors, under the present very well-known operative systems.

Finally, are detailed three types of applications to complex systems: a net model and two related to meteorological phenomena (hail and fires), which their antecedents were both research projects in academic ambits of the Comahue National University (Neuquén, Argentina).

2. ESTADO DEL ARTE

2.1 ALEATORIEDAD

El desarrollo en forma rigurosa de la noción de aleatoriedad puede conducir a cuestiones complejas que se alejan del significado simple y común a todos los idiomas que conceptualizan a la aleatoriedad, (François, 1992, p.24), como aquella "condición de un fenómeno cuya ocurrencia no obedece a alguna secuencia causal observable".

Esta noción de aleatoriedad suele ser aplicada a los sucesos futuros y en otros casos a sistemas que tienen un cierto grado de coherencia y permanencia, resultando por tal razón limitada.

La idea de aleatoriedad "a lo largo de la historia ha recibido diferentes interpretaciones, ninguna satisfactoria, y la definición estadística no siempre resulta convincente" (Botanero C., 2001). Dos aspectos se pueden distinguir en la idea de aleatoriedad: el proceso de generación, que matemáticamente se conoce como experimento aleatorio; y el patrón de la secuencia aleatoria, producida como consecuencia del experimento.

Hacia el año 1965 A.N. Kolmogorov y G. Chaitin (Chaitin, 1975, pp.50-52), en trabajos independientes, proponían una definición de sucesión de números aleatorios equivalente a la siguiente proposición:"Una sucesión aleatoria es una sucesión que no puede ser especificada por un algoritmo más corto que la propia sucesión".

Como ejemplo (Ríos Insua, 1997, pp.14-16), la sucesión 0010010010...se interpreta como no aleatoria, puesto que se puede dar una descripción algorítmica más corta que la propia sucesión.

De acuerdo a la definición anterior se arriba a una paradoja (Botanero C., 2001): por un lado, si la sucesión no se puede codificar en forma más abreviada, implica la ausencia de patrones periódicos como su característica esencial; y por otra parte, el enfoque estadístico establece que un fenómeno es aleatorio si se ajusta al cálculo de probabilidades.

De la discusión de estas ideas se podría introducir en la definición, un modelo computacional similar a la máquina de Turing, con limitaciones en el tiempo y en el espacio empleados para realizar sus cálculos.

Ríos Insua propone la siguiente definición: "Una sucesión de números es aleatoria si nadie que utilice recursos computacionales razonables en tiempo razonable

puede distinguir entre la serie y una sucesión verdaderamente aleatoria de un forma mejor que tirando una moneda fiel para decidir cuál es cuál".

Otro punto de vista asociado a la aleatoriedad lleva a discusiones sobre l impredecibilidad y a resultados de carácter paradójico.

D.H. Lehmer (1951) y J. N. Franklin (1962) dieron desde la teoría matemátic de la probabilidad los fundamentos de la definición rigurosa de "sucesión aleat ria", que más recientemente D. E. Knuth (Knuth,1981, pp. 152-155) y H. Niede rater y Tichy desarrollan a partir de las definiciones de sucesiones y subsucesione infinito-distribuidas.

A los fines de nuestro trabajo llamaremos a la sucesión $\{u_n\} = u_1, u_2$,. de números reales, en el rango $0 < u_i < 1$, una sucesión de números aleatoric cuando los u_i "se comporten como si en realidad fuesen aleatorios"; es decir, qu satisfagan las condiciones estadísticas de independencia y uniformidad, aunqu por su naturaleza sean determinísticos.

En otros términos, dada una función de distribución F, con valores en el con junto R de los números reales, se generará una sucesión de números reales qu simulan una sucesión de variables aleatorias independientes y que son distribuida idénticamente según la distribución F.

2.2 GENERACION DE NUMEROS ALEATORIOS

Los métodos de generación de sucesiones de números aleatorios se han desarrollado conjuntamente con los estudios sobre simulación por Monte Carlo. Históricamente se inician con los métodos tipo "dados" (Birger Jansson, pp.22-38); luego las "tablas" de distinto origen y cantidad de dígitos aleatorios; la generación a través de números "trascendentes" (e , π); y la generación mediante "algoritmos", durante las últimas décadas.

La difusión de los microcomputadores dio nuevo impulso a la investigación de metodologías orientadas a generar "números aleatorios" en forma numérica.Esta alternativa de generación por medio de fuentes numéricas internas es la más empleada en el presente.

Estos métodos son secuenciales en el sentido de que cada nuevo número de la sucesión ha sido generado por uno o más de los elementos precedentes de la sucesión.

Su mayor divulgación se debe a que son rápidos, eficientes y producen sucesiones que pueden ser reproducidas.

El Cap.3 está dedicado a presentar algunos de los métodos congruenciales programados en computadoras usando algoritmos determinísticos para generar sucesiones de números, también de carácter determinístico. Por lo tanto, dichas sucesiones, que sólo se aproximan a verdaderas sucesiones aleatorias, se denominaran sucesiones de números pseudoaleatorios, y los generadores que las producen se llamarán Generadores de Números Pseudoaleatorios (GNPA).

En muchos casos, en la literatura específica, una vez realizadas estas precisiones, se designan Generadores de Números Aleatorios (GNA), en vez de GNPA.

Entre los más recientes estudios sobre los métodos GNA, deben destacarse varias líneas de investigación:

• Los Generadores de Fibonacci retardados (Lagged Fibonacci Generators, LFG)

Son una generalización de los métodos congruenciales. Marsaglia (1995) demuestra que con una buena selección de la semilla y de la operación binaria que se usa en la fórmula de recurrencia, se logran modelos eficientes. La caracterización del período máximo "está bien estudiada y se basa en el análisis de sucesiones lineales recursivas de enteros" (Ríos Insua y otros, 1996), de la forma:

$$x_i = (a_1 x_{i-1} + a_2 x_{i-2} + ... + a_r x_{i-r}) (\mathrm{mod}\, 2^n)$$

• Generadores paralelos

Con las denominadas super-computadoras, los métodos de Monte Carlo experimentaron un particular desarrollo de los algoritmos que emplean múltiples procesadores en paralelo.

Las ideas básicas en este campo dieron lugar a varias alternativas técnicas en función de las arquitecturas paralelas: La utilización de un único GNA y una o varias semillas, predeterminadas ó aleatorias; y el uso de varios GNA separados que se asignan independientemente a cada procesador. En ambos casos no se alcanzaron hasta la fecha resultados completamente satisfactorios para problemas donde se han comprobado fuertes correlaciones entre distintas sucesiones de número pseudoaleatorios.

• Los Generadores no lineales

Se caracterizan por no utilizar estructuras reticulares, como son los congruenciales lineales. Como lo sostiene P. L'Ecuyer (cfr.41) esta tendencia puede modificarse introduciendo no-linealidad en un generador a través de dos caminos:

1) Usando un generador con función de transición lineal T, produciendo la salida mediante una transformación G, no lineal del estado.

2) Construir un generador con una función de transición T no lineal.

Una propiedad común de los generadores no lineales es que producen una estructura que generalmente se representa por un hiper plano que contiene, a lo sumo una cantidad t de $t - uplas$ solapadas de números.

Los generadores del tipo 1) se denominan "Generadores Congruenciales de Inversión Explícita" (ICG, Eichenhauer y Lehn, 1986).

Dada la fórmula de recursíon de primer orden $y_{n+1} \equiv t(y_n) \,(\mathrm{mod}\ m)$, donde n=0,1,2,3,...; y t una función periódica con valores enteros en el conjunto {0,1,..., m-1}, de forma tal que los valores de t y y_0 se elijan para que la sucesión $\{y_0, y_1, ...\}$, resulte periódica, con un período no superior a m. Es usual que el módulo sea primo y una potencia de 2. El método congruencial así definido parte de una simple clase de funciones t, define inversos multiplicativos y usa aritmética

modular para obtener la sucesión de números enteros así definida: $x_n = y_n/m$, para n=0,1,2,...

Estos generadores tienen propiedades estructurales muy interesantes, como el cálculo del período máximo; sin embargo en opinión de L'Ecuyer quedan pendientes investigaciones futuras que cubran la actual falta de experiencias prácticas en el uso de estos generadores.

Otros generadores no lineales son los creados para uso específico en criptografía, como los indicados en el Cuadro N° 2.2 en las referencias NIST y HENKOS.

Una referencia final muy interesante (cfr.Pierre L'Ecuyer), en el mismo Cuadro N° 2.2 ya citado, es acerca de un generador congruencial lineal del tipo "múltiple generador recursivo" que se recomienda especialmente, puesto que posee un período máximo igual a $m = 2^{191}$, ideado por L'Ecuyer, Simard, Chen y Kelton (2001), y que forma parte del paquete RNG package y está codificado en Java, C y C++. En Operation Research, Vol. 47 (1), 1999, L'Ecuyer presenta el generador MRG 32k3a en su artículo "Good Parameters and Implementations for Combined Multiple Recursive Random Number Generators", y los resultados obtenidos en arquitecturas de 64 bits (SUN Ultra-2 y DEC AlphaStation 250).

2.3 PRUEBAS DE ALEATORIEDAD

Entre las numerosas pruebas concebidas para probar la aleatoriedad de las sucesiones de números pseudoaleatorios, se eligen aquéllas que han sido más usadas y que mejor se adaptan a su programación en computadores.

La gran cantidad de pruebas (tests) se debe a que hay muchas formas en que una sucesión de tales números puede fallar en su aleatoriedad.

Sin embargo, como lo expresa Knuth (D. Knuth, 1981, p.38) si una sucesión se comporta como aleatoria para los tests $T_1, T_1, ..., T_n$, no se puede asegurar en general que para T_{n+1} no se encuentre una falla.

En la práctica, las baterías de pruebas usuales van de 5 a 10 pruebas, aproximadamente.

En cuanto al número de elementos de la sucesión $\{x_n\}$ es también variable. Bates y Zirkle (1971) aplicaron varias pruebas a sucesiones de 50 mil números. Learmonth y Lewis (1973), tomaban 6 millones 500 mil números de varias sucesiones.

Dudewicz y Ralley (E. Dudewicz y Th. Ralley, 1981, p.17), utilizan sucesiones de 10 mil números cada una, hasta totalizar diez millones.

Como lo señala P.Hellekalek del pLAB Project (cfr.Cuadro N°2.2, continuación), hay dos hechos concernientes a los GNA que deben destacarse:

1) La falta de garantías de un GNA de su rendimiento en la práctica numérica, ya que sólo pueden hacerse predicciones de su comportamiento. Los números pseudoaleatorios generados, al transformarse en variables aleatorias, no siempre ajustan satisfactoriamente cualquier distribución dada. En alguna aplicación, aún los generadores más confiables pueden fallar.

2) Aunque no haya garantías sobre un generador, existen herramientas matemáticas seguras para analizar los resultados erróneos provenientes de un generador inapropiado.

Al someter al generador a las Pruebas Teóricas, la mayor dificultad, y la más importante es el *análisis de correlación*.

Una larga experiencia muestra que ciertas pruebas teóricas permiten hacer predicciones muy confiables sobre el rendimiento de un generador cuando se analizan las muestras producidas en las Pruebas Empíricas (estadísticas). Pero hasta el presente no ha sido posible encontrar una vinculación de carácter matemático

entre las pruebas teóricas y los rendimientos empíricos de las muestras producida por un GNA.

En las investigaciones en curso realizadas en pLAB Proyect se consideran d mayor importancia las siguientes pruebas teóricas: Discrepancia, Prueba Espec tral y Prueba Espectral Ponderada.

H. Niedereiter (cfr.54) define teóricamente a la *discrepancia* como una medid cuantitativa de la desviación de una sucesión con respecto a una distribució uniforme.

Las Pruebas Empíricas, para el citado grupo de la Universidad de Salzburg son indispensables pues se sostiene con acierto que toda prueba empírica es un simulación, y por lo tanto si se elige cuidadosamente una prueba en particula entonces se puede cubrir una amplia clase de problemas de simulación.

Al contrario de las pruebas teóricas, una característica de las pruebas empírica es que los GNA son tratados como cajas negras haciendo posible la observació de los rendimientos para distintas partes del período.

Las pruebas empíricas informadas coinciden con algunas del sistema Gentstp que se detallan en el Cap.4 (prueba de las corridas y prueba de los pares seriales

Otra batería de pruebas estadísticas que se incluye en el Cuadro 2.2 con la re ferencia NIST es la biblioteca DIEHARD (Marsaglia G.,"The Marsaglia Rando Number CD-ROM including the DIEHARD BATTERY OF TESTS OF RAN DOMNESS", Florida State University 1997, *ftp://stat.fsu.edu/pub/diehard)* qu contiene 15 pruebas, las cuales pueden correrse bajo los sistemas: Dos, Unix Linux. Las denominaciones originales se muestran en el Cuadro N°2.3.

2.4 OTRAS APLICACIONES

La necesidad de números pseudoaleatorios se encuentran en muchas aplicaciones criptográficas, por ejemplo, las claves utilizadas en sistemas criptográficos; los protocolos de criptografía que requieren entradas de tales números para generar firmas digitales y su autenticación.

La comprobación estadística y su relación con el criptoanálisis es también tema de discusión.

Estas pruebas pueden ser útiles como un primer paso para determinar si un generador es conveniente o no para una aplicación criptográfica en particular.

Desde 1997, un grupo técnico para la generación de números aleatorios (RNG-TWG, National Institute of Standards and Technology - N.I.S.T, U.S. Commerce Department's Technology Administration, *http://csrc.nist.gov/rng*) ha desarrollado una batería de pruebas estadísticas apropiadas para la evaluación de generadores de números pseudoaleatorios.

La publicación del NIST 800-22, "A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications", describe una colección de 16 pruebas en total, cuyas denominaciones originales son:

Frequency (Monobits) Test; Test for Frequency within a Block;Runs Test; Test for the Longest Run of Ones in a Block; Random Binary Matrix Rank Test; Discrete Fourier Transform (Spectral) Test; Non-overlapping (Aperiodic) Template Matching Test; Overlapping (Periodic) Template Matching Test; Maurer's Universal Statistical Test; Lempel-Ziv Complexity Test; Linear Complexity Test; Serial Test; Approximate Entropy Test; Cumulative Sum (Cusum) Test; Random Excursions Test; Random Excursions Variant Test.

Una nueva versión de la colección de pruebas estadísticas (9/12/2004, versión 1.7) se ha desarrollado para el uso de plataformas basadas en PC, incluyendo los siguientes cambios:

Reemplazo de Lempel-Ziv Test; Actualización de valores en el Spectral Test; verificación de parámetros adicionales antes de invocar cada prueba estadística del paquete, entre otros.

2.5 CONCLUSION

Teniendo en cuenta el "Estado del Arte" descrito en los parágrafos precedentes, nuestro trabajo puede inscribirse en dos líneas bien definidas: la generación de números pseudoaleatorios y las pruebas teóricas y estadísticas que conforman el sistema GENTSTPC, desarrollado en el Cap.4.

Para los GNA se han programado algoritmos congruenciales lineales y otros, compatibles con las arquitecturas elegidas, PC con longitud de palabra-memoria de 32 bits, bajo los sistemas operativos más frecuentemente utilizados (DOS y Windows 9x) en ámbitos académicos de nuestras Universidades latinoamericanas.

En este sentido, nuestro trabajo es el primero ofrecido, bajo estas restricciones, en el mundo. Otros intentos similares no han sido completados en países tales como Hungría (Levendovszky), Finlandia (C. Lin) y Bélgica (Van der Meulen).

En relación con las Pruebas Estadísticas de Aleatoriedad, a las clásicas Pruebas Empíricas, hemos añadido las denominadas Pruebas de Entropía, y una Metodología apropiada para guiar al usuario en la selección de un generador eficiente (cfr. 4.3.3).

Este importante aspecto de las Pruebas, fue complementado con una verificación experimental de una simulación aplicada a un caso real donde se confirmaron las tendencias históricas de un fenómeno natural (Tormentas de Granizo en la zona del Alto Valle de Río Negro y Neuquén, Argentina), al obtener resultados satisfactorios para las variables aleatorias estudiadas, las cuales fueron representadas en el proceso mediante varios generadores eficientes, incorporados simultáneamente en la simulación.

Cuadro N° 2.2 Otros sistemas de GNA y Pruebas Estadísticas

REFERENCIA	DESCRIPCION	WEB (HTTP://)
ENT	Paquete pruebas estad.: entropía, χ^2,	www.fourmilab.ch/
	medias arit., cálculo □ por Monte Carlo,	random/
	corr. serial.	
	S.O: Unix y D.O.S	
Mersenne	Generador muy rápido. Aut.: Matsumoto	www.math.keio.ac.jp/
Twister	y Nishimura; período:$2^{19937}-1$; ACM	~matumoto/emt.html
	Trans. Vol. 8 N°1, 1998	
NIST	Statistical Test Suite	www.csrc.nist.gov/rng/
(National	Batería de pruebas estadísticas en suces.	rng2.html
Institute of	binarias de Nros. pseudoaleat. para usos	
Standards and	criptográficos.(ver. 1.7, Dic. 2004)	
Technology -USA)	S.O: Unix y Windows	
HENKOS	GNA para uso en claves criptográficas	www.crypto.8k.com
	Módulo 1: preparación de semillas;	
	Módulo 2: generación suces. Nros	
	pseudoaleat (origen: Rumania).	
	Ha superado las pruebas estad.: FIPS,	
	ENT, DIEHARD y NIST	
	S.O: D.O.S, Windows y Linux	
Netlib	Biblioteca de GNA en código fuente	netlib.org/random/
	C y Fortran; incluye RANLIB	
NHSE	Guía de información sobre GNA;	www.npac.syr.edu/
	programas disponibles en National	projects/random/
	HPCC Software Exchange (NHSE)	
	para computadores de alto rendimiento	
	y prueba de aleatoriedad de dichos	
	generadores.Syracuse University (USA)	
GNU Scientific	Biblioteca(GSL); rutinas en C, ANSI C,	sources.redhat.com/
Library	ver. GLS 1.6, 31/12/2004 para C.Num	

Cuadro Nº 2.2 Otros sistemas de GNA y Pruebas Estadísticas (continuación)

REFERENCIA	**DESCRIPCION**	**WEB** (HTTP://)
pLAB Project	Servidor con información sobre teoría	random.mat.sbg.ac.at/
WWW.VirtualLibrary	y práctica (GNA).	
	Varios paquetes de programas en	
	ANSI-C, C++ y Smalltalk para generar	
	y evaluar Nros. pseudoaleat.	
	Director: P. Hellekalek-Univ.Salzburgo.	
Random Number	Taygeta Scientific Inc.; Información	www.taygeta.com/
Generation	y artículos sobre GNA	random.html
Pseudo random	Biblioteca de normas de GNA	www.agner.org/
number generator	eficientes, con códigos en C++ y	random/
	Lenguaje Ensamblador.	
	S.O: Windows, Linux, BSD (sist. de	
	32 bits),Ult. ver. Julio 2004.	
	Aplic. Monte Carlo.	
RngPack	Paquete de GNA en Java, ult. ver.	www.honeylocust
	Noviembre 2003.	.com/RngPack/
Random Number	"Computational Sc. Education	csep1.phy.ornl.gov/
Generators	Project"(CSEP, 1995).Libro	
	electrónico para enseñanza Cs.de la	
	Comp. e Ing.	
Luc Devroye:	Vínculos: Artículos, publicaciones y	cgm.cs.cn/
	programas.	
Random Number	McGill University, Montreal Canadá	
Generation	GNA eficientes: Artículos y paquetes	www.iro.umontreal.ca
Pierre LEcuyer	de programas- Univ. of Montreal	/~lecuyer/papers.html

Cuadro N° 2.3 Batería de pruebas Diehard

1.	Birthday Spacings Test
	Esta prueba provee además de información a Kstest.
2.	The Overlapping 5-Permutation Test (Operm5)
	Esta versión utiliza 2'000'000 de enteros.
3.	Binary Rank Test Para Matrices 31x31
4.	Binary Rank Test Para Matrices 32x32
5.	Binary Rank Test Para Matrices 6x8
6.	The Bitstream Test
7.	The Test Opso, Opso And Dna
8.	The Count-The-1's Test
9.	A Parking Lot Test
	Provee información de entrada a KStest.
10.	The Minimum Distance Test
11.	The 3dspheres Test
12.	The Sqeeze Test
13.	The Overlapping Sums Test
14.	The Runs Test
15.	The Craps Test

3. METODOS DE GENERACION Y PRUEBAS ESTADISTICAS

3.1 METODOS DE GENERACION

Se reconoce como una condición necesaria para que un generador sea de buena calidad el hecho de obtener resultados satisfactorios como resultado de la aplicación de una batería de pruebas estadísticas a una parte sustancial de la sucesión de números que se han generado.

La calidad de los números pseudoaleatorios generados está referida a la propiedad de "aparecer" como variables aleatorias, distribuidas uniformemente, de modo que toda sucesión de ellas sea estadísticamente independiente en el intervalo de cero a uno.

Por otra parte, tal calidad se relaciona con propiedades relativas a la eficiencia computacional, tales como, mínimo almacenamiento en memoria del computador, portabilidad y sencillez de programación del generador.

Para A. Law y W.D. Kelton (Law y Kelton, pp. 423-424) un "buen" generador debe tener estas otras propiedades: brindar repetibilidad de una misma sucesión a efectos de ser usada en la simulación de diferentes sistemas que se comparan; permitir la separabilidad de la sucesión de números en distintos lotes para diferentes propósitos en un mismo sistema de simulación.

Algunos de los requerimientos computacionales que inicialmente eran de una considerable importancia, en el presente van perdiendo significación frente a la popularización de modernos microcomputadores que han avanzado en eficiencia por sus mayores capacidades de memoria real y virtual, y la velocidad de cálculo, entre otras propiedades.

3.1.1 Los métodos de congruencias lineales

Entre los más usuales métodos de generación de números pseudoaleatorios, los métodos congruenciales son los que mejor satisfacen los criterios requeridos para las sucesiones de números pseudoaleatorios. Fueron introducidos por Lehmer (1951).

Expresión general:

$$x_{i+1} = a\, x_i + c \pmod{m}$$

donde x_i, a, c y m son números enteros no negativos que verifican además:
m, módulo $m > 0$
a, factor multiplicativo, $0 \leq a < m$
c, constante aditiva, $0 \leq c < m$
x_0, valor inicial o semilla, $0 \leq x_0 < m$
Cuando $c \neq 0$ el generador se denomina Congruencial Mixto.

La sucesión $\{x_i\}$ permite definir la sucesión $\{u_i\}$ de números reales uniformemente distribuidos entre 0 y 1, mediante los cocientes: $u_i = \frac{x_i}{m}$

El período del generador, igual a la longitud de un ciclo, es a lo sumo igual al módulo m.

Si el período es exactamente m, se dice que el generador tiene un período completo o máximo, por lo que la elección de la semilla será arbitraria entre los números 0,1,2,..., (m–1).

A los fines de las aplicaciones en simulación de modelos hay por lo tanto interés en que m sea un valor tan grande como la arquitectura del procesador del computador lo permita.

Algunos resultados de la teoría de números imponen condiciones al factor multiplicativo a y a la constante aditiva c con el propósito de lograr un período completo.

Teorema 1

Dada la sucesión generada por:
$x_{i+1} = a\, x_i + c \pmod{m}$, con $i = 0, 1, ...$ Si $mcd\ (a, m) = 1$, entonces, existe un entero positivo d llamado período de la sucesión, tal que $x_0 = x_d$

Teorema 2

Si $mcd\,(a, m) = 1$, entonces el período de la sucesión generada por:
$x_{i+1} = a\, x_i + c \pmod{m}$ es $d = m$, si y sólo si para todo divisor primo p de m se tiene:

(i) $a = 1 \pmod{p}$ si p es un factor primo impar
(ii) Si 4 es divisor de m, entonces $a = 1 \pmod{4}$

(iii) $mcd\ (c, m) = 1$

Teorema 3

Si en la sucesión $x_{i+1} = a\, x_i + c \pmod{m}$ se tiene $c = 0,\ m = 2s$, para $s \geqslant 3$ $a = 3 \pmod 8$, entonces, su período es $d = 2^{s-2}$, si x_0 es un entero impar.

Muchos generadores congruenciales mixtos utilizan como módulo potencias de dos, por ejemplo $m = 2^{31}$ ó 2^{32}.

Las condiciones del Teorema 1 son satisfechas cuando c es impar y a es una potencia de dos, más uno.

En particular, los llamados números de Mersenne son de la forma 2^{s-1}

(s = 2,3,5,7,13,17,19,..., 20'996'011, 24'036'583), y suelen tomarse como valores del módulo $m = 2^{s-1}$ por la propiedad de ser primos.

El último número (2^{s-1}), es el 41°, fue descubierto recientemente por Findley J., Woltman y Kurowski, (http://www.mersenne.org/prime.htm); posee alrededor de 7.2 millones de cifras decimales, y para verificar la condición de primo se emplearon distintas computadoras con un tiempo de cálculo variable entre 5 y 14 días.

Los generadores congruenciales con la constante aditiva $c = 0$ se denominan Generadores Congruenciales Multiplicativos:

$$x_{\,i+1} = \ a\ x_{\,i} \pmod{m}$$

En esta clase de generadores la obtención de períodos grandes impone condiciones al factor multiplicativo a que debe ser, para tal fin, de la forma $8t \pm 3$, para un valor cualquiera de t ; y la semilla x_0 debe ser un número impar.

Knuth suministra otras condiciones: el módulo m debe ser primo de la forma $2^{31} - 1$, por ejemplo.

En el caso de microcomputadores, con palabras-memoria de 32 bits, es sabido que el máximo número positivo que puede almacenar es 2^{32} -1 = 4'294'967'295, pero al representar números negativos se reduce la magnitud, de 2^{32} -1 a 2^{31} -1 ya que se destina un bit para el signo. El número máximo representable será entonces 2^{31} -1 = 2'147'483'648

Los generadores multiplicativos con módulo m igual a una potencia de 2 son más sencillos de programar.

Suponiendo que se dispone de un microcomputador que usa 32 bits incluyendo al bit de signo, elegimos como módulo $m = 2^{31}$.

Luego de efectuar la multiplicación: $x_{i+1} = a\, x_i$ deseamos conservar los 31 bits menos significativos.

Si ese producto no supera una cadena de 31 bits, los bits menos significativos serán correctos, sin embargo, el bit situado más a la izquierda, durante la multiplicación, será considerado como un bit común, pero en todos los casos en que su valor final sea un "1", por la convención del signo, el producto será considerado negativo, procediendo a la conversión de (-x) por la técnica del complemento "a dos". La técnica del complemento a 2, en binario, aplica el complemento a 1, y después la adición de un 1. Los números negativos son almacenados entonces así: $(2^{32}$ - 1 - x) + 1.

Aplicado al caso en que $x_{i+1} < 0$, se definirá:

$$x_{i+1} = [(2^{32} - 1) - x_{i+1}] + 1$$

La representación de números enteros negativos mediante el "complemento a uno", se forma con los complementarios a uno de los bits del entero positivo correspondiente, y fue usada por procesadores de grandes equipos como CDC en las décadas del 60 y 70.

3.1.2 Métodos de combinación de generadores

Estos métodos llamados también compuestos combinan dos o más generadores para definir un nuevo generador compuesto con el fin de obtener mejores rendimientos en el comportamiento estadístico de las sucesiones de números pseudoaleatorios.

M.D. Mc Laren y G. Marsaglia (1965) propusieron un método en el que se determina cuál de los k números ya generados debe ser el próximo en la sucesión. En 1968 W.J. Westlake (Naylor, 1971, pp.396-397) mejora el anterior en cuanto al consumo de espacio de memoria.

En 1968 Marsaglia y T. A. Bray elaboraron un método compuesto que incluye tres generadores congruenciales y su programación en FORTRAN para los equipos IBM 360/ 7094 y para UNIVAC 1108.

Entre otros aportes más recientes pueden citarse los de K. Marse y S.D. Roberts (1983), J-P L'Ecuyer (1988) y L'Ecuyer - Côté (1990).

3.1.3 Métodos de desplazamiento de registros

Operan directamente con los bits para formar los números pseudoaleatorios. Aparecen como una extensión a órdenes k $\geqslant$ 1 de los generadores congruenciales del tipo descrito en 2.1.1

Para m primo una expresión de esta clase es la siguiente:

$$x_{i+1} = a_1 x_{i-1} + a_2 x_{i-2} + \ldots + a_k x_{i-k} \pmod{m}$$

de donde se obtiene la sucesión $\{u_i / \ u_i = \frac{x_i}{m} \}$

La idea básica es emplear como módulo a números primos pequeños que faciliten su posterior implementación computacional, por ejemplo $m = 2$. De acuerdo a los trabajos de R.C. Tausworth el método se denominó de Recurrencia Lineal Módulo Dos.

Según la teoría del Algebra Finita cuando m es primo podemos encontrar los coeficientes:

$a_1, a_2, \ldots, a_k$ tales que la sucesión $\{ x_i \}$ tenga un período de longitud $m^k - 1$.

Para que ello se verifique es una condición necesaria y suficiente que los coeficientes a_i sean los mismos que los de un polinomio P(z), primitivo según módulo m:

$$P(z) = z_k - a_1 z_{k-1} - \ldots - a_k$$

Según H. Niederreiter (1992, pp.191-204) hay dos métodos de este tipo: el digital de múltiples etapas y el método FSR y GFSR (generalized feedback shift register, T.G. Lewis y W. H. Payne, 1973).

Los generadores de la clase FSR (feedback shift register) se basan en trinomios del tipo:

$$T(x) = x^p + x^q + 1 \ , \ con \ q < p$$

Un generador de este tipo podrá ser considerado como un generador de sucesiones de bits aleatorios.

Cada nuevo bit b[k] podrá determinarse por la relación de recurrencia:

$$b[k] = b[\ k - p + q] \ XOR \ b[k - p]$$

donde XOR es el operador lógico OR exclusivo.

Los generadores del tipo GFSR incluyen ventajas como la producción de números en forma multidimensional, poseer un período independiente de la palabra-memoria del computador y portabilidad del algoritmo.

3.2 GENERADORES CON POLINOMIOS DE WEYL

Estos generadores fueron estudiados en particular por Altaber (J.L. Altaber, Annales de la Faculté des Sciences, Université de Clermont, Número 34, 1967)

3.2.1 Sucesiones equidistribuidas

Se plantea calcular un número N de términos de r sucesiones $\{x_n\}$ que son equidistribuidas e independientes en el espacio G_k formado por los números cuya mantisa posee un número finito de k cifras, del tipo:

$$0\,.\,a_1 a_2 \ldots a_k$$

donde: $0 \leq a_i \leq g-1\ ;\ i = 1, \ldots, k$ (g es una base numérica)

Las sucesiones $\{x_n\}$ formadas por estos números del espacio G_k y que verifican ciertas propiedades se denominan equidistribuidas o independientes, según las siguientes definiciones.

Sucesión equidistribuida [$d1$]

$\{x_n\}$ es equidistribuida en G_k si para todo $a \sqsubset G_k$

si $\dfrac{N'(N\,;\ x_i = a)}{N} \rightarrow \dfrac{1}{g^{\,k}}$ cuando $N \rightarrow \infty$

N' designa el número de i $\leq N$ para los cuales $x_i = a$

Sucesiones independientes [$d2$]

Las r sucesiones equidistribuidas en G_k representan r sucesiones independientes de una variable aleatoria con ley uniforme en (0,1).

Las r sucesiones $\{x_n^j\}$, $j = 1, 2, \ldots, r$ equidistribuidas en G_k son independientes en G_k si cualesquiera sean los r números k_l de G_k se verifica, cuando $N \rightarrow \infty$:

$$\frac{N'(N\ ;\ x_i^1 = k_1\ ,\ x_i^2 = k_2\ ,\ldots,\ x_i^r = k_r\ )}{N} \rightarrow \frac{1}{g^{\,kr}}$$

Sucesiones con términos independientes $[d3]$

La independencia p a p de los términos de una sucesión equidistribuida en G_k, se define seguidamente para un p determinado.

Dada una sucesión $\{ x_n\}$ de G_k, se dice que sus términos son independientes p a p si para todo j < p, y cualesquiera sean los números k_j de G_k, se cumple:

$$\frac{N'(N\ ;\ x_i = k_1\ ,\ ,\ \ldots\ , x_{i+j-1} = k_j\ )}{N} \rightarrow \frac{1}{g^{r\,j}}$$

cuando $N \rightarrow \infty$

Un camino para construir tales sucesiones se encuentra a partir de los teoremas de H.Weyl.

Las siguientes definiciones se refieren a sucesiones finitas e infinitas de puntos.

Sucesiones infinitas

Dado el hipercubo unitario $C = [0,1]^r$ del espacio R^r al cual pertenecen los puntos $P_1, P_2, \ldots$ de una sucesión y D un dominio interior a C de volumen v, se dice que la sucesión $\{P_i\}$ es equidistribuida en C si, para cualquier dominio $D \subset C$, se tiene:

$$\frac{N'(N\ ;\ P_i\ \square D\ )}{N} = v$$

Sucesiones finitas

Sea una sucesión finita de puntos:

$$P_n = \{x_{n_1}, x_{n_2}, \ldots, x_{n_r}\}$$

donde $1 \leq n \leq N$, pertenecientes al hipercubo C.

A fin de caracterizar la repartición de tal sucesión en el hipercubo C se define el concepto de **discrepancia** de una sucesión finita de puntos.

Sea $N'(\alpha\ ,\beta)$ el número de puntos P_q $(q \leq N)$ que son interiores al paralelepípedo $\mathcal{P}\ (\alpha_j \leq x_{q_j} < \beta_j)$ de volumen

$$\prod_{j=1}^{r} (\beta_j, \alpha_j)$$

siendo α y β dos puntos del espacio R^r de coordenadas α_j, β_j, con $0 \leq \alpha_j < \beta_j < 1;\ 1 < j \leq r$

Se llama discrepancia de la sucesión $\{P_n\}$ a la cantidad:

$$D_n = sup_{(\alpha,\beta)} \left| \frac{N'(\alpha,\beta)}{N} - \prod_j (\beta_j - \alpha_j) \right|$$

Propiedades

1) Para cualquier N se verifica: $0 \leq D_n \leq 1$

2) La sucesión $\{P_n\}$ es equidistribuida en C, si D_n tiende a cero, cuando N tiende a infinito.

3) Para caracterizar la distribución de una sucesión finita se selecciona, entre las sucesiones disponibles, aquélla que para un N dado tiene la menor discrepancia.

3.2.2 Construcción de sucesiones

Sucesiones equidistribuidas

Para construir una sucesión equidistribuida $\{\widehat{x}\}$ partiendo de otra $\{x\}$, que es equidistribuida en (0,1), se tiene en cuenta el siguiente Lema.

Lema 1

Si $\{x_n\}$ es una sucesión equidistribuida en (0,1), entonces la sucesión $\{\widehat{x}\}$ que se obtiene reemplazando cada término x_i de $\{x_n\}$ por su aproximación racional de orden k, es equidistribuida en G_k.

El teorema 1 permite la construcción de una sucesión equidistribuida e independiente.

Teorema 1

Dada $\{x_n\}$ equidistribuida en (0,1), entonces la sucesión $\{\underbrace{L\,x_n}\}$ es equidistribuida en (0,1), para todo entero L$\neq 0$, siendo $\underbrace{L\,x_n} = L\,x_n - [L\,x_n]$

Designando con el símbolo $\widehat{x_n}$ a la aproximación racional de orden kr del n-ésimo término de la sucesión $\{x_n\}$, se tiene:

$$\widehat{x_n} = 0.\alpha_n^1 \alpha_n^2 ... \alpha_n^{kr} \qquad [1]$$

$$0 \leq \alpha_n^i \leq g-1 \ , \ i = 1, ..., kr$$

Por el Lema 1, la sucesión [1] es equidistribuida en G_k.

Luego, si se agrupan las cifras de $\widehat{x_n}$, en paquetes de k cifras, dicho desarrollo puede expresarse:

$$\widehat{x_n^j} \ = \ 0.\alpha_n^{(j-1)k+1} \alpha_n^{(j-1)k+2} ... \alpha_n^{jk} \ ; \ j = 1, ..., r \qquad [2]$$

Lema 2

Las r sucesiones $\widehat{x_n^j}$ definidas en [2] son equidistribuidas e independientes en (0,1).

Del teorema 2 se puede inferir que si $\{x_n\}$ es equidistribuida en (0,1), entonces la sucesión $\underbrace{\{x_n g^{k(j-1)}\}}$ es equidistribuida en (0,1)

Por el Lema 1 la sucesión $\{\widehat{x_n^j}\}$ obtenida tomando las aproximaciones racionales de orden k de los términos de la sucesión $\{x_n g^{k(j-1)}\}$ resulta equidistribuida en G_k.

Para demostrar la independencia de las sucesiones $\{\widehat{x_n^j}\}$ definidas en [2], se considera que, cualesquiera sean los r números k_i de G_k, la definición de $\widehat{x_n^i} = k_i \, (i = 1, ..., r)$ implica que:

$$\widehat{x_n} = a = \sum_{i=1}^{r} k_i \, g^{(i-1)k} \quad , \, a \, \square G_{kr}$$

En consecuencia, se tiene:

$$\frac{N'(N \, ; \, x_n^i = k_i \ , \ i = 1, ..., r)}{N} = \frac{N'(N \, ; \, \widehat{x_n} = a)}{N} \qquad [3]$$

de lo que se infiere que en el límite, cuando N tiende a infinito, el cociente [3] tiende al valor $\frac{1}{g^{k\,r}}$

Entonces, por [d3] las sucesiones [2] son equidistribuidas e independientes.

Teoremas de Weyl

En el estudio de las sucesiones equidistribuidas en (0,1), H. Weyl ha proporcionado dos teoremas que además establecen condiciones necesarias y suficientes para que una sucesión de puntos pertenecientes al hipercubo C sea equidistribuida en C.

Teorema 4

Para que una sucesión de puntos $\{P_n\}$ sea equidistribuida en el hipercubo C de un espacio de dimensión finita, es necesario y suficiente que para toda función $f(P)$, integrable en el sentido de Riemann en C, se verifique :

$$\int_C f(P)\,dw = \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N} f\,(P_n)$$

donde dw es el diferencial de volumen en el espacio.

Teorema 5

Para que la sucesión $\{P_n\}$ sea equidistribuida en el hipercubo C es condición necesaria y suficiente que para todo vector L con coordenadas enteras, no todas nulas, se cumpla:

$$\lim_{L\to\infty} \frac{1}{N} \sum_{n=1}^{N} e^{2\,i\,n\,<L,\,P_n>} = 0$$

donde el producto escalar del vector L por el vector P_n se designa $< L, P_n >$.

3.2.3 Algoritmo generador de sucesiones

El siguiente teorema 6, de Weyl, define procedimientos explícitos para construir sucesiones equidistribuidas en (0,1).

Teorema 6 (Weyl)
Si $\varphi(t) = A_0 t^m + A_1 t^{m-1} + ... + A_{m-1} t + A_m$
es un polinomio de grado $m \geqslant 1$, en el que al menos un coeficiente distinto de A_m, es irracional, entonces la sucesión $\{\underbrace{\varphi(n)}\}$ es equidistribuida en (0,1).

Un estudio sobre la discrepancia de estas sucesiones muestra que aquellas sucesiones de primer grado del tipo $x_n = \underbrace{An}$, donde A es un número irracional, poseen una discrepancia más baja y no exigen de A una buena representación racional.

También se muestra que dicha aproximación racional es menos buena a medida que los cocientes incompletos de su desarrollo en fracción continua son más pequeños.

Designando $\mathcal{B}$ la clase de números A cuyos cocientes incompletos son acotados, el resultado siguiente es válido para sucesiones de primer grado del tipo $\{\underbrace{An}\}$:

$$\lim_{N \to \infty} \left| \frac{N D_n}{\log N} \right| < k$$

donde D_n es la discrepancia; k es un número real positivo; $A \square \mathcal{B}$

Si se utiliza como polinomio con $m = 1$ se tendrá un monomio del tipo:

$$\varphi(I) = \underbrace{AI}$$

lo cual tiene la ventaja de simplificar el cálculo al reducirse a un algoritmo de adiciones puesto que :

$$\varphi(I+1) = \underbrace{A(I+1)} = \underbrace{A + \varphi(I)}$$

El problema entonces se plantea como el cálculo de N términos de r sucesiones equidistribuidas e independientes en G_k empleando el polinomio de Weyl:

$$\varphi(I) = \underbrace{AI}$$

El teorema 6 (Weyl) permite definir un procedimiento de construcción de sucesiones equidistribuidas, y luego empleando los resultados, por Lema 2, obtener sucesiones independientes.

Los datos requeridos por el algoritmo son:

- Disponer de una aproximación racional $\widehat{A}$ de un número irracional A (v.gr. $\square, e$)

- Una adecuada elección de $\widehat{A}$ en función de la base numérica g; el número k de cifras de los términos de la sucesión a construir, la cantidad r de sucesiones independientes y el número N de términos de las r sucesiones.

Como el número $\widehat{A}$ es de longitud, en general mayor que la longitud de la palabra-memoria del computador, con problemas de desbordamiento, es más interesante trabajar en la misma escala que la palabra del microcomputador, para lo cual se ha adoptado el monomio:

$$\varphi(I) = \underbrace{AI}$$

La adición se programa en la base $B = g^{k-1}$

donde k es la cantidad de cifras y g la base numérica.

Se define M como el entero más próximo que verifica la siguiente relación entre los datos k y N.

$$M \geqslant \frac{\log N}{k}$$

Ejemplo:

N =10000
r = 1000
k = 10
M = 1

El número irracional elegido para una primera aplicación es $A = \square$, cuya aproximación racional es:

$$\widehat{A} = 0.a_1 a_2 \ldots a_u$$

donde el número de cifras, después del punto decimal, es el orden de la aproximación racional de $\square$, y se denota u:

$$u = (m + r)\, k$$

Para el ejemplo: $u = (1 + 1000)10 = 10010$

La determinación de diez mil diez cifras decimales se ha logrado utilizando una función evaluadora en un lenguaje orientado al cálculo numérico-simbólico (MAPLE V).

Luego se han programado las conversiones necesarias a archivos de caracteres compatibles con un lenguaje de alto nivel (Fortran), el empleado en la creación del sistema GENTSTPC, a fin de codificar el algoritmo de generación de números pseudoaleatorios.

La estructura adoptada para expresar AI es un arreglo:

$$A(I) = a_{(I-1)k+1}\, a_{(I-1)k+1} \ldots a_{I\,k}$$

el cual para el ejemplo producirá las cifras:

$I = 1$ $\quad a_1 \quad a_2 \quad \ldots \quad a_{10}$

$I = 2$ $\quad a_{11} \quad a_{12.} \quad \ldots \quad a_{20}$

... $\quad$ ...

$I = 10001$ $\quad a_{1001}\ a_{1002}\ \ldots\ a_{10010}$

El modelo computacional que genera los números pseudoaleatorios partiendo del archivo conteniendo las cifras de la aproximación racional de A se muestra seguidamente en pseudocódigo.

1. Inicio algoritmo POLIWEYL
2. Ingreso datos:

2.1 Archivo $(m + r)\,k$ cifras aproximación racional en $a(i)$

2.2 Cantidad k de cifras de los números a generar

2.3 Valor de la constante m

2.4 Valor de r, cantidad de sucesiones independientes

2.5 n, número de términos de cada sucesión

3. Para $indi = 1\,,\ n$ hacer:

$d = 0 \quad ; \quad j = m + r + 1 \quad ; \quad q = 10^{k-1}$

Mientras $j \geqslant 1$ hacer:

$j = j - 1$

$x(j) = a(j) + x(j) + d$

si $x(j) \leq q$ entonces $d = 0$; sino $d = 1$

$x(j) = x(j) - q$

FinSi

FinMientras

Grabar archivo: $numreg = float(x(j)) \,/\, 10^{10}$

FinPara

4. Fin algoritmo

En forma similar, cambiando el ingreso en 2.3 del algoritmo, por una aproximación racional de otro número irracional, $A = e = \exp(1)$, se obtienen otras sucesiones de números pseudoaleatorios.

Finalmente, fusionando ambas aproximaciones racionales, de □ y e, en orden monótono creciente, y empleando el mismo algoritmo, se determinan nuevas sucesiones. En los tres casos, es posible aplicar las pruebas estadísticas, con los resultados que se detallan en 4.2.4

3.3 PRUEBAS ESTADISTICAS

Entre las numerosas pruebas concebidas para probar la aleatoriedad de las sucesiones de números pseudoaleatorios, se eligen aquéllas que han sido más usadas y que mejor se adaptan a su programación en computadores.

La gran cantidad de pruebas (tests) se debe a que hay muchas formas en que una sucesión de tales números puede fallar en su aleatoriedad.

Sin embargo, como lo expresa Knuth (D. Knuth, 1981, p.38) si una sucesión se comporta como aleatoria para los tests $T_1, T_2, ..., T_n$, no se puede asegurar en general que para T_{n+1} no se encuentre una falla.

En la práctica, las baterías de pruebas usuales van de 5 a 10 pruebas, aproximadamente.

En cuanto al número de elementos de la sucesión $\{x_n\}$ es también variable. Bates y Zirkle (1971) aplicaron varias pruebas a sucesiones de 50 mil números. Learmonth y Lewis (1973), tomaban 6 millones 500 mil números de varias sucesiones.

Dudewicz y Ralley (E. Dudewicz y Th. Ralley, 1981, p.17), utilizan sucesiones de 10 mil números cada una, hasta totalizar diez millones.

3.4 PRUEBAS EMPIRICAS Y TEORICAS

Una clasificación de las pruebas de los números pseudoaleatorios, las considera agrupadas en Pruebas Empíricas y Pruebas Teóricas.

Pruebas Empíricas son aquéllas donde se calculan ciertos estadísticos de las sucesiones $\{u_i\}$ a fin de analizar estadísticamente cuan próximos están a variables aleatorias independientes y uniformemente distribuidas en (0,1).

Como existen varias maneras en que $\{u_i\}$ puede desviarse en aleatoriedad, hay varios tipos de estas pruebas.

En 4.2.2.3 y A.2 (Anexo I) se dan los detalles de varias pruebas y algunos ejemplos de bibliotecas disponibles.

Las pruebas teóricas más utilizadas son cuatro: correlación serial, potencia-período del generador, factorización y la prueba espectral. Son las pruebas denominadas por D. Knuth (Knuth, 1981) "tests matemáticos", desarrollados en el anexo A.3

3.5 PRUEBAS BASADA S EN ENTROPIA

En esta sección se estudian y aplican pruebas de entropía para analizar propiedades de uniformidad en la evaluación de generadores de números aleatorios.

Otros aportes recientes a los métodos de entropía se han realizado en las Universidades: Syracuse University (Dept. of Mathematics); Katholieke Universiteit Leuven (Dept. of Mathematics); y la Université de Montréal (Département d'Informatique et de Recherche Opérationnelle).

3.5.1 Definiciones y teoremas

Un estimador de entropía fue propuesto por Vasicek (1976) y estudiado por Van der Meulen y Dudewicz (1981), quienes mostraron cómo puede ser usado para evaluar si una muestra aleatoria proviene de la distribución uniforme [0,1] y también desarrollaron evaluaciones asintóticas y por Monte Carlo de los puntos porcentuales de la distribución del estimador.

La **entropía** diferencial (o entropía) de una variable aleatoria X con función de densidad f , se define como:

$$H(X) \equiv H(f) = -\int_{-\infty}^{\infty} f(x) \log f(x) dx \qquad [1]$$

y tiene las siguientes propiedades:

- Si $X \square [0,1]$ y tiene convergencia con probabilidad uno, $P(X \in [0,1]) = 1$, entonces:

$$H(f) \leq 0 \qquad [2]$$

- Entre todas las funciones de densidad concentradas en $[0,1]$ la uniforme f_0 maximiza H(f):

$$H(f_0) = 0 \qquad [3]$$

Definición 1: Dos funciones de densidad f_1 y f_2 se dicen entrópicamente distinguibles si $H(f_1) \neq H(f_2)$.

Definición 2: Una función de densidad f^* se dice entrópicamente única (o *e-única*) en una clase $\mathcal{C}$ de densidades si

$f\ ^*\square\mathcal{C}\ \ ,\ \nexists f\ \square\mathcal{C}\ \ ,\ \ f \neq f^*$ tal que $H(f) = H(f*)$.

Por la propiedad [3] sabemos que en la clase $\mathcal{C}$ de densidades en $(0,1)$, f_0 es *e-única*.

Como en general la función de densidad de un generador es desconocida, en consecuencia no es posible calcular el verdadero valor de $H(f)$. En su lugar se estudian algunos estadísticos que operan con muestras de sucesiones $\{X_i\}$ de números pseudoaleatorios, producidos por ese generador.

Sea $X_1, ..., X_n$ una muestra aleatoria de una distribución F absolutamente continua con función de densidad f, y sean $X_{(1)}, ..., X_{(n)}$ los valores ordenados de $X_1, ..., X_n$.

El estimador de $H(f)$ (Vasicek 1976) es:

$$H_{m,n} = n^{-1} \sum_{i=1}^{n} \log \left\{ \frac{n}{2m} \left[X_{(i+m)} - X_{(i-m)} \right] \right\} \qquad [4]$$

donde $1 \leq m < n/2$, $X_{(\,j\,)} = X_{(1)}$ para $j < 1$; $X_{(j)} = X_{(n)}$ si $j > n$

Dudewicz y Van der Meulen (1981) propusieron una prueba de uniformidad basada en la e-unicidad de f_0 que se describe a continuación:

Sea $X_1, ..., X_n$ una muestra aleatoria de una distribución F absolutamente continua con función de densidad f concentrada en $[0,1]$.

Sea f_0 la función de densidad de $U(0,1)$.

A nivel α , la prueba que rechaza $H_0 : f = f_0$ puede sustituirse por:

$H_A : \ f \neq f_0$ si y sólo si:

$$H_{m,n} \leq H^*_\alpha(m,n) \qquad [5]$$

donde $H^*_\alpha(m,n)$ es el punto percentil 100α de la distribución de $H_{m,n}$ bajo la hipótesis de uniformidad.

Por la e-unicidad de f_0 entre todas las densidades concentradas en $[0,1]$ y por la consistencia del estimador, se deduce que la prueba propuesta es consistente contra todas las alternativas f en $[0,1]$.

Dudewicz y Van der Meulen demuestran que si f se concentra en $[0,1]$ entonces, con una probabilidad de convergencia igual a uno, $H_{m,n} \leq 0$

Teorema : Bajo H_0, si $m = O(n^{\frac{1}{3}-\delta})$, $\delta > 0$,entonces las cantidades:

$$(6mn)^{\frac{1}{2}} \left[H_{m,n} - \log\left(\frac{n}{2m}\right) + \log(n+1) + \gamma - R(1, 2m-1) \right] \qquad [6]$$

$$(6mn)^{\frac{1}{2}} \left[H_{m,n} + \log(2m) + \gamma - R(1, 2m-1) \right] \qquad [7]$$

son asintóticamente $N(0,1)$ cuando $n \to \infty$

Aquí, γ es la constante de Euler, $\gamma = 0.57721566490$ y, para j $\leq$ k, se tiene:

$$R(j,k) = k^{-1} + (k-1)^{-1} + ... + j^{-1}$$

En particular: $R(1,k) = k^{-1} + (k-1)^{-1} + ... + 2^{-1} + 1, \quad k \geqslant 1$

Usando [6] tenemos que:

$$\hat{H}^*_{\alpha}(m,n) = \Phi^{-1}(\alpha)(6mn)^{\frac{-1}{2}} - \log(2m) - \gamma + R(1, 2m-1) - \log\left(\frac{n+1}{n}\right) \qquad [8]$$

es una aproximación asintótica al punto percentil 100α de la distribución de $H_{m,n}$; $\Phi(\cdot)$ es la función de distribución normal estándar.

3.5.2 Evaluación basada en rangos de percentiles asintóticos

Usaremos aquí la relación [4] para evaluar si una muestra de números aleatorios, que se supone uniforme, tiene entropía consistente con la uniformidad.

Obtendremos en primer lugar $H_{m,n}(G)$ a partir de una sola muestra de números pseudoaleatorios suministrados por el generador G, a evaluar; en este caso escogemos nueve generadores: URN01, URN02, URN03, URN10, URN11, URN13, URN22, URN36 y URN39.

Fijamos $n = 10000$ y tomamos catorce valores de $m = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 30, 40$

Los valores de $H_{m,10000}$ se calculan para los $n = 10000$ números aleatorios de la muestra, después de ser ordenados en forma monótona creciente, utilizando la fórmula [4] aplicada a los 14 valores de m.

Los percentiles asintóticos de los estadísticos $H_{m,n}$, expresados en términos de su distribución asintótica bajo H_0 (hipótesis de uniformidad), se grafican en las Figuras N° 3.5.2.1 (URN01 y URN10), N° 3.5.2.2 (URN13 y URN22) y N° 3.5.2.3 (URN39).

Los gráficos seleccionados muestran un comportamiento común de esos generadores, con excepción de URN10, en cuanto a que los puntos asintóticos para un nivel de significación $\alpha = 0.05$ corresponden al punto 5% de la distribución asintótica nula, de lo que se infiere que los valores $H_{m,n}$ deben alcanzar valores porcentuales mayores que 5.0 para que la hipótesis de uniformidad no sea rechazada.

Entonces, todos los generadores estudiados, excepto URN10, superan esta prueba de entropía para "pequeños" valores de m ($m < 15$) y fallan para algunos $m \geqslant 15$.

Estas interpretaciones se basan en una única muestra de diez mil elementos, y por lo tanto debe ser tomada como una ilustración del uso de $H_{m,n}$ para comparar generadores de números aleatorios y no como una prueba definitiva.

3.5.3 Algoritmo para la evaluación de generadores según la distribución asintótica de un estimador de entropía

La expresión del punto percentil de la distribución de $H_{m,n}$ está dada por [10]:

$$100 \times \Phi\left(\sqrt{6mn}\,(c + \log(2m) + \gamma - R(1, 2m-1))\right) \qquad [10]$$

Teniendo en cuenta que la distribución del estadístico $H_{m,n}$ no es conocida, es posible considerar el estudio de la correspondiente distribución asintótica, bajo la hipótesis H_0 de uniformidad, mediante la determinación de los rangos de los percentiles dados por la relación [10], donde $c = H_{m,n}$.

Si tal distribución de los $\mathrm{H}_{m,n}$, que designaremos $H^*_\alpha(m,n)$, puede ser acotada superiormente por una distribución normal $N(0,1)$, para n tendiendo a infinito, entonces puede darse una aproximación asintótica al punto de percentil 100α de la distribución $H_{m,n}$; elegiremos $\alpha = 0.05$ y algunos valores de m, con n fijo.

Dicho estimador se designará $\widehat{H}^*_\alpha(m,n)$ y se define en la expresión [11]:

$$\widehat{H}^*_\alpha\,(m,n) = \Phi^{-1}(\alpha)(6mn)^{-1/2} - \log(2m) - \gamma + R(1,2m-1) - \log\left(\frac{n+1}{n}\right) \qquad [11]$$

El siguiente algoritmo fue concebido para evaluar muestras de números pseudoaleatorios en base a sus propiedades de entropía, considerando aquellos valores de $H_{m,n}$ obtenidos por el estimador de Vasicek, [4], y que satisfacen la prueba de Dudewicz y V. der Meulen, [5], para catorce valores de m y $l = 1000$ muestras producidas por el generador G, cada una conteniendo $n = 10000$ números pseudoaleatorios.

Posteriormente, se calculan los rangos de los percentiles P_m de la distribución asintótica $\widehat{H}^*_\alpha(m,n)$, [11], para un nivel de significación $\alpha = 0.05$, y para los valores: $m = 1, 2, ..., 9, 10, 15, 20, 30, 40$.

Mediante la relación [12] se determinan los valores promedio de los percentiles para el total de muestras analizadas.

$$P_m = 100 \times \Phi\,(\frac{F_m}{l}) \qquad [12]$$

siendo:

$$F_m = 100 \times \sqrt{6mn}\,(\,c + \log(2m) + \gamma - R(1, 2m-1)) \qquad [13]$$

ALGORITMO

1) Asignar cantidad de lotes y cantidad de números pseudoaleatorios por cada lote: $l = 1000$; $n = 10000$

2) Para $i = 1, ...l$ hacer:

2.1 Para $j = 1, ..., n$: generar U_j
FinPara

2.2 Para $m = 1, 2, ..., 9, 10, 15, 20, 30, 40$ calcular:

$$C_{m,i} = \frac{1}{n} \sum_{i=1}^{n} H_{m,n,i}$$

2.3 Calcular : F_m ([13])
2.4 Determinar: $\widehat{H}^{*}_{\alpha}(m,n)$ ([11])
2.5 Comprobar la condición: $H_{m,n} \leq \widehat{H}^{*}_{\alpha}(m,n)$
FinPara
FinPara

3) Calcular rangos de los percentiles de la distribución asintótica del estimador $H_{m,n}$ expresado por P_m.
Para $m := 1, 2, ..., 9, 10, 15, 20, 30, 40$ calcular: $P_m = 100 \times \Phi\,(\frac{F_m}{l})$ ([12])
siendo $\Phi\,(.)$ la función de distribución normal estándar.

4) Aplicación del criterio de aceptación del generador G

Para $m = 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20$

si $5 \leq P_m \leq 50$,
entonces, G satisface la prueba de entropía. FinPara

Habiendo analizado los generadores URN01, URN02, URN03, URN10, URN11, URN13, URN22, URN36 y URN39, sólo URN10 no supera el criterio 4) del precedente algoritmo, como tampoco satisface el conjunto de pruebas empíricos estadísticos de GENTSTPC.

3.5.4 Evaluaciones basadas en pruebas de ajuste a distribuciones asintóticas

Una prueba de bondad de ajuste del estadístico $H_{m,n}$ a una distribución normal asintótica suele plantearse para un lote de $N = 100$ valores muestrales de $H_{m,n}$ para $m = 1, 2, 3, 4$; $n = 10000$.

La hipótesis nula es que cada una de las muestras de $H_{m,n}$ proviene de una distribución normal dada por [14], es decir, que la distribución $N(\square, \square^2)$ es, para $n \to \infty$, $m = O(n^{\frac{1}{3}-\delta})$, $\delta > 0$:

$$N\left(-\log(2m) - \gamma + R(1, 2m-1), \frac{1}{6mn}\right) \quad [14]$$

Para realizar Chi-cuadrado hemos extendido la prueba a 1'000 lotes de 10'000 números cada lote, de manera de poder disponer de mayor cantidad de muestras del estimador estudiado.

En el método se utilizan diez intervalos equiprobables:

$$(-\infty, x_1]\,;(\,x_1, x_2]\,;\ldots;.(x_9, \infty)$$

donde los valores de la función de distribución acumulada N (0,1) son:

$$F(x_i) = \frac{i}{10} \quad ; \quad i = 1, ..., 9$$

Los puntos extremos de los intervalos son respectivamente los valores de:

$$\Phi^{-1}(F(x_i))$$

que deben transformarse según los parámetros de $N(\square, \square^2)$, de [14], para cada

$m = 1, 2, 3, 4$, mediante la expresión:

$$x_i = \square_i \Phi^{-1}(F(x_i)) + \square_i$$

El estadístico:

$$\chi^2 = \sum_{i=1}^{10} \frac{(o_i - e_i)^2}{e_i}$$

se calcula mediante una de las subrutinas de un paquete de programas (TSTENTR) donde $e_i = 10$ es la cantidad esperada de valores H_{mn}; y o_i la cantidad de valores observados en cada intervalo.

Como se tienen $\nu = (10 - 1)$ grados de libertad y $\alpha = 0.05$, resulta:

$$\chi^2 < \chi^2_{9,\,0.95} = 16.9190$$

Considerando 10 lotes de muestras, cada una conteniendo 100 valores de $H_{m,n}$, se constata que en una proporción del 70 al 100%, los generadores URN01, URN02, URN03 y URN11, para $m = 1, 2, 3, 4$, verifican la prueba, y por consiguiente la hipótesis H_0 no es rechazada.

En el Cuadro N° 3.5.4.1 se indican estos resultados para los mismos generadores de números pseudoaleatorios del parágrafo anterior.

Los mismos coinciden con algunos de los generadores que han superado el algoritmo del parágrafo precedente 3.5.3, pero no concuerdan en su totalidad, debido a que en ese caso el estimador H_{mn} fue sometido a otras pruebas, como son los tests de Vasicek y Dudewicz-Van der Meulen.

3.5.5 Evaluaciones basadas en entropía para distribuciones discretas

En recientes trabajos sobre pruebas de uniformidad basadas en formas de entropía discreta, se estudia la calidad de los generadores congruenciales lineales, en particular, (L' Ecuyer, Compagner y Cordeau, 1996), con importantes recursos numéricos de cálculo, como el procesamiento en paralelo.

La entropía de una variable discreta X, con valores en un conjunto discreto S, y probabilidad $p_x = P\ [X = x]$ se define:

$$H_d = -\sum_{x \in S} p_x \log_2\ p_x \qquad [15]$$

Sea una cadena (string) de L bits aleatorios independientes, con igual probabilidad de obtener "1" ó "0", entonces la cantidad total C de resultados que se puede lograr con los L bits es $C = 2^L$.

Cada cadena de longitud L tiene la misma probabilidad de ocurrencia igual a $2^{-L} = 1/C$.

Si se considera una aplicación entre el conjunto de cadenas binarias y el conjunto de números enteros positivos, se establece una correspondencia entre cada variable entera discreta X y una cadena de bits, de manera que X posee una distribución discreta sobre el conjunto $S = \{0, 1, 2, ..., C-1\}$.

Denominamos N_x al número de veces que un valor x es obtenido cuando se ha tomado una muestra de n variables aleatorias, $X_1, X_2, ..., X_n$, independientes, por hipótesis.

Definición: Estimador de entropía discreta o empírica.

$$\widehat{H}_{\,d}\,(C, n) = -\sum_{x=0}^{C-1} \frac{N_x}{n}\,\frac{\log_2 N_x}{n} \qquad [16]$$

La construcción de una cadena de bits producida por un número real u_i, perteneciente a $(0, 1)$, se puede plantear a partir de la expresión binaria fraccionaria de ese número, considerando su primeros k bits, con una determinada precisión igual a l, tal que:

$$u_i = \sum_{j=1}^{l} b_{ij} 2^{-\,j}$$

donde b_{ij} es una cifra binaria obtenida como la "parte entera" del producto entre la parte fraccionaria resultante y 2, iterando el método hasta el orden l .

Dada una sucesión de números pseudoaleatorios producidos por un generador congruencial lineal, la hipótesis es que la sucesión:

$$\{b_{ij}\} = \{b_{11}...b_{1l};\ b_{21}...b_{2l};\ b_{31}...b_{3l};\ ...\}$$

está formada por bits aleatorios independientes, cada uno asumiendo el valor 0 ó 1, con una probabilidad igual a 1/2.

Para aplicar el algoritmo propuesto por L'Ecuyer-Compagner-Cordeau es necesario fijar previamente un número n de subcadenas disjuntas a escoger en la sucesión $\{b_{ij}\}$, como así también su longitud l, en bits.

Luego, calcular N veces la entropía empírica dada por [16], con substrings disjuntos, de manera de obtener una sucesión de esos valores:

$$\{T_i\} = \{T_1, T_2, ..., T_n\}$$

Dos caminos se plantean a continuación para verificar la hipótesis nula:

- Hallar una distribución empírica de la $\{T_i\}$, para luego ser comparada con la respectiva distribución teórica de $\widehat{H}_d$

- Verificar si hay una correlación significativa entre los sucesivos valores de la entropía discreta, es decir entre los pares (T_i, T_{i+1})

Seguidamente desarrollamos una prueba de correlación para verificar la hipótesis H_0 a fin de comprobar si los pares $(T_i, T_{i+1}),\ 1 \leq i \leq N-1$, están correlacionados de una manera significativa. Si así fuese, ello significaría que una baja(alta) entropía en una parte de la sucesión $\{b_{ij}\}$ tiende a ser seguida de una baja (alta) entropía en la parte consecutiva de la misma.

En la comparación de los elementos $\{T_i\}$ de las sucesiones, fueron consideradas otras tres variantes posicionales alternativas a los pares formados con elementos consecutivos: (T_j, T_{m+j}) ; (T_j, T_{m+j-10}) y (T_j, T_{j+10}), donde m es el punto medio de la sucesión.

Por otra parte, utilizando los modelos teóricos que proporciona la Estadística, es posible sustituir al estadístico T_i por una expresión más simple, dada por los pares (S_i, S_{i+1}), definidos de acuerdo a la siguiente proposición.

Proposición:

Bajo la hipótesis H_0 se tiene:

$$E\,[\,\widehat{H}_{\,d}(C,n)\,]\;\;=\;\;-C\;\sum_{j=0}^{n}\frac{j}{n}\log\left(\frac{j}{n}\right)\binom{n}{j}\frac{(C-1)^{n-j}}{C^n}\qquad[17]$$

$$Var\,[\,\widehat{H}_{\,d}(C,n)\,]\;=C\sum_{j=0}^{n}\left(\frac{j}{n}\log\left(\frac{j}{n}\right)\right)^2\binom{n}{j}\frac{(C-1)^{n-j}}{C^n}+$$

$$+C(C-1)\sum_{j=0}^{n}\sum_{k=0}^{n}\frac{j}{n}\log\left(\frac{j}{n}\right)\frac{k}{n}\log\left(\frac{k}{n}\right)\binom{n}{j}\binom{n-j}{k}\frac{(C-2)^{n-j-k}}{C^n}-$$

$$-\;E^2\,[\,\widehat{H}_{\,d}(C,n)\,]\qquad[18]$$

$$S\;(C,n)\;=\frac{\widehat{H}_d(C,n)\;-\;E\,[\,\widehat{H}_d(C,n)\,]}{\sqrt{Var\,[\,\widehat{H}_d(C,n)\,]}}\quad\text{tiende a}\;\;N\,(0,1)\,\text{cuando}\;n\to\infty$$

para C fijo.

En consecuencia, el estadístico $\widehat{H}_{\,d}(C,n)$ puede reemplazarse por el estimador T_i en la expresión anterior:

$$S_i=\frac{T_i-E\,(T_i)}{\sqrt{Var(\,T_i\,)}}$$

La correlación entre los S_i es simplemente:

$$\hat{\square}_N=\frac{1}{N-1}\;\sum_{i=1}^{N-1}S_{\,i}\,S_{i+1}$$

Bajo la hipótesis H_0, cuando N tiende a infinito, entonces el coeficiente de correlación converge a cero, con probabilidad uno.

Una prueba estadística se deduce entonces:

Calcular $\sqrt{N}\,\hat{\square}_N$, para un valor grande de N, entonces H_0 se rechaza si ese valor no está próximo a cero.

Como aplicaciones de esta prueba de Correlación de Entropía Discreta a sucesiones de números pseudoaleatorios se seleccionaron los mismos generadores congruenciales analizados en los parágrafos precedentes.

Como precisión finita y longitud de las subcadenas se fijaron, respectivamente $k = l = 10$ y $k = l = 12$, ya que valores mayores implican tiempos de procesamiento muy superiores.

Algunos de los principales resultados logrados a través de la ejecución de un paquete de programas confeccionado a este fin, se detallan en el Cuadro N° 3.5.5.1

Para $l = 12$, la convergencia a valores cada vez más pequeños, se verifica en URN11, URN02, URN03, URN36, URN13, URN22 y URN01; mientras que en el caso de URN10 y URN39 la tendencia es hacia valores cada vez más grandes.

En correspondencia con las pruebas Chi-cuadrado realizadas en el caso de entropía continua, para el caso discreto se analizaron también diez lotes de 100 valores del estadístico $\widehat{H}_d\,(C, n)$, en los casos l =10, $l = 12$, para C=2^l , n =10'000.

La hipótesis nula expresa que cada una de las cien muestras proviene de una distribución normal N (□, □^{2}), donde la media y la varianza están dadas respectivamente por [17] y [18].

Los resultados del Cuadro N° 3.5.5.2 reflejan comportamientos análogos, para $l = 12$, en cuanto a los Generadores estudiados, con la salvedad de URN22, que registra seis de diez verificaciones de la prueba $\chi^2 < \chi^2_{9,\,0.95} = 16.9190$, y los mismos rechazos de la hipótesis nula, respecto a los generadores URN10 y URN39.

El hecho de que los restantes generadores analizados no superen esta prueba estadística, pero que sean satisfactorios para un conjunto de pruebas empíricas tan amplio como GENTSTPC, salvo URN10, demuestra que la uniformidad en entropía no puede ser usada como una metodología única y aislada.

Cuadro N° 3.5.4.1 Entropía continua (Prueba Chi-cuadrado)

Estimador de entropía H_{mn}	M=1 N° Lot.	M=2 N° Lot.	M=3 N° Lot.	M=4 N° Lot.	Resultados $\chi^2_{9,\,0.95} = 16.919$
URN01	7	10	7	8	Satisfactorio
URN02	9	9	8	6	Satisfactorio
URN03	10	9	8	9	Satisfactorio
URN10	0	0	0	0	No satisfactorio
URN11	7	9	9	8	Satisfactorio
URN13	0	2	7	5	No satisfactorio
URN22	6	6	6	6	No satisfactorio
URN36	0	0	3	3	No satisfactorio
URN39	0	0	0	0	No satisfactorio

Cuadro N° 3.5.5.1 Entropía discreta (Prueba de correlación)

		(S_j, S_{m+j}) $\hat{\square}$	(S_j, S_{m+j-10}) $\hat{\square}$	(S_j, S_{j+10}) $\hat{\square}$	(S_j, S_{j+1}) $\hat{\square}$	Tendencia promedio
URN01	L=10	1.89566	0.88262	0.16994	4.87122	1.9
	L=12	2.26544	0.05509	0.17853	1.99545	1.1
URN02	L=10	0.59679	-0.53661	0.31009	3.65172	1.0
	L=12	-0.04278	-0.05928	0.14159	1.99521	0.5
URN03	L=10	1.43194	0.52214	-0.15174	4.23430	1.5
	L=12	0.46247	0.22679	-0.07062	-0.09549	0.5
URN10	L=10	2141.3	209.8	209.8	4278	1709.8
	L=12	2921.8	286.3	286.3	5838	2333
URN11	L=10	2.66596	0.298061	0.42629	6.71774	2.5
	L=12	2.73529	-0.11533	-0.27152	-1.37052	0.2
URN13	L=10	0.98366	-0.02222	0.08951	3.27527	1.0
	L=12	1.551027	0.50008	-0.16003	1.54438	0.9
URN22	L=10	3.60180	0.86672	-0.00465	5.92911	2.6
	L=12	2.12835	0.25557	-0.33061	2.21113	1.0
URN36	L=10	1.94630	0.33413	0.40584	2.20992	1.2
	L=12	0.99311	0.04464	0.40939	1.191756	0.7
URN39	L=10	189147	17917	21646	453301	170502
	L=12	4150434	398118	425182	88118663	3446399

Cuadro N° 3.5.5.2 Entropía discreta (Prueba Chi-cuadrado)

Lotes		1	2	3	4	5	6	7	8	9	10	Resultados $\chi^2_{9,\,0.95}$ =16.919
URN01	L10	13.4	17.4	16.2	23	30.6	8.2	14.8	23.8	53.8	31.2	No satisf.
	L12	22.6	9.8	10	13.2	15	15.6	10	15.4	23.6	8.2	Satisf. 80%
URN02	L10	29.2	21.4	25.4	10	16.8	17.4	18.6	14.8	26.6	15.4	No satisf.
	L12	21.6	10.6	15	3.6	9.2	17	12.2	21.2	6.6	8.4	Satisf. 80%
URN03	L10	25.2	26.2	19.4	18.2	33.4	18.8	16	34.6	11.8	28.8	No satisf.
	L12	21.4	8.4	5.8	12.2	14.8	19.6	13.8	8.2	10	7.6	Satisf. 80%
URN10	L10	900	900	900	900	900	900	900	900	900	900	No satisf.
	L12	900	900	900	900	900	900	900	900	900	900	No satisf.
URN11	L10	32.6	30.6	19.8	30.4	21	19	20.6	30.6	15.4	60	No satisf
	L12	3.6	12.8	10	9.2	14.2	5.2	9.8	11.4	14.2	10.6	Satisface
URN13	L10	21.4	17.6	28.8	11.6	16.8	36.4	19.6	26.8	12.6	11.4	No satisf
	L12	11.6	12.4	22.8	15.4	4.6	18.8	13	8.2	9.8	14.6	Satisf. 80%
URN22	L10	25.4	12.2	19.4	35	37.4	40.2	30.4	13.2	28.2	31.6	No satisf.
	L12	18.4	18.8	13.4	19.2	5.8	10.4	16	17.4	17	27.8	No satisf.
URN36	L10	12.2	14.8	18.8	23	17.8	14	33.4	25.4	15	12	No satisf.
	L12	22.8	10.6	18.4	19.6	14.6	12	10	22.8	12.8	12.6	No satisf.
URN39	L10	643	900	900	900	900	900	900	900	900	900	No satisf.
	L12	880	900	900	900	900	900	900	900	900	900	No satisf.

Figura N° 3.5.2.1 Percentiles asintóticos H_{mn} (URN01 y URN10)

Figura N° 3.5.2.2 Percentiles asintóticos H_{mn} (URN13 y URN22)

Figura N° 3.5.2.3 Percentiles asintóticos H_{mn} (URN39)

resultados que se detallan en 4.2.4

4. SISTEMA GENTSTPC

La concepción y diseño del sistema GENTSTPC tiene su origen en la necesidad de disponer de una biblioteca de programas para generar y probar sucesiones de números pseudoaleatorios empleando equipos y redes de microcomputadores de amplia utilización en diversos medios académicos y tecnológicos. De esta especial orientación a las máquinas PC deriva precisamente la denominación del sistema: GEN(generación)TST(tests)PC(microcomputadores).

El desarrollo de Gentstpc implica, por una parte, el estudio y la determinación de un nuevo conjunto de generadores congruenciales para producir sucesiones de números pseudoaleatorios (cfr.4.1), teniendo en cuenta los actuales equipos informáticos, los nuevos lenguajes de programación y los sistemas operativos de las PC, replanteando aspectos computacionales en versiones existentes,(cfr. Testrand, Anexo I), y determinando nuevos algoritmos de Generadores de Números Aleatorios, GNA (en rigor, pseudoaleatorios, pero por convención usaremos la abreviatura GNA en lugar de GNPA).

Por otro lado, se analizan las bibliotecas de pruebas estadísticas teóricas y empíricas más conocidas, destinadas a la verificación de la calidad de las sucesiones de números pseudoaleatorios creados por los GNA, y se introducen nuevas Pruebas (v.gr.entropía continua), para verificar la calidad estadística de las sucesiones generadas (cfr.4.2).

Es necesario destacar la orientación y el estímulo, en estos aspectos particulares del trabajo, de uno de los creadores de Testrand, el Dr. Edward J. Dudewicz, (Syracuse University, New York), a través de intercambios académicos personales en Neuquén y Syracuse, que posibilitaron avanzar y profundizar en los aspectos técnicos.

En orden a establecer una nueva metodología para la selección de un GNA eficiente (cfr.4.3), se analizan las pruebas empíricas basadas en uniformidad, y se introducen los contrastes basados en un estimador de entropía, con el complemento de una prueba teórica cuando los GNA son congruenciales lineales; y finalmente, se confirma la calidad de los GNA seleccionados, a través de resultados satisfactorios logrados en una aplicación a un proceso de simulación, con el funcionamiento simultáneo e independiente de varios GNA.

El paquete de programas Gentstpc, que incluye 40 generadores y 16 pruebas estadísticas, ha sido codificado en Fortran para el sistema operativoWindows, totalizando cerca de 18'000 líneas. Los subprogramas se distribuyen en 131 subrutinas y 6 subprogramas funciones.

La última versión prevé tres opciones principales diferentes: realizar pruebas estadísticas empíricas, teóricas y pruebas de entropía en sucesiones de números creados internamente; las mismas tareas, en sucesiones de números pseudoaleatorios ingresadas mediante archivos generados por otros métodos externos al paquete; y generar únicamente archivos de datos con sucesiones de números creados por los generadores de Gentstpc.

En el Anexo II (CD-ROM) se encuentran todos los programas fuente del sistema, como así también, el Manual del Usuario, los programas de entrada de datos, y un archivo de datos con números pseudoaleatorios correspondientes a un generador (URN13), para ser empleado, como ejemplo, en las pruebas estadísticas.

4.1 LOS GENERADORES

Los generadores fueron seleccionados teniendo en cuenta su amplia utilización y sus particulares características satisfactorias frente a las pruebas estadísticas, y su mejor adaptación a las arquitecturas de microcomputadores con palabras de 32 bits.

Al conjunto de treinta generadores congruenciales lineales ya conocidos, se agregan otros diez, los cuales constituyen los más recientes aportes al "estado del arte".

4.1.1 Algoritmos

Conservando anteriores notaciones, se los designará URN31,URN32,...,URN40.

En el Cuadro N° 4.1.1 se muestra una breve descripción del origen del generador, incluyendo a los treinta primeros, el lenguaje fuente en que fue escrito y eventualmente los nombres genéricos usados para describirlos en la literatura publicada.

URN31, URN32 y URN33

Estos generadores fueron estudiados por H. Miyazaki (1987) con las características que en cada uno se indica:

URN31

$$x_{i+1} = 2456949\ x_i \,(\mathrm{mod}\ 2^{24})$$

Semilla igual a 1234567 y período igual a 4194304

URN32

$$x_{\,i+1} = 3513383\ x_{\,i}\ (\ \mathrm{mod}\ 2^{25}\ )$$

Semilla igual al de URN31 y período igual a 8388608

URN33

$$x_{\,i} = y_{\,2i+2}$$

siendo:

$$y_{2i+1} = 11257\ y_{2i} \pmod{10^8 + 7}$$

$$y_{2i+2} = 1253\ y_{2i+1} \pmod{10^8 + 37}$$

Semilla $y_0 = 12345678$, período 9103417

URN34

Un generador combinado que para el uso en microcomputadores fue propuesto por F. Sezgin (1987). Emplea tres generadores multiplicativos definidos por las tres sucesiones:

$$w_{i+1} = 44\ w_i \pmod{2039}$$

$$y_{i+1} = 45\ y_i \pmod{2037}$$

$$z_{i+1} = 41\ z_i \pmod{2003}$$

Semillas: $w_0 = y_0 = z_0 = 1$
Designando a los tres módulos : $m_1 = 2039$; $m_2 = 2037$; $m_3 = 2003$
la sucesión $\{u_i\}$ se obtiene después de efectuar una permutación en tres tablas:

$u_i = w_i/m_1 - y_i/\ (m_1 m_2) + z_i\ /\ (m_1 m_2 m_3)$

Un código Fortran lo proporcionan Karian y Dudewicz (Z. Karian y E. Dudewicz, 1991, p.120)

URN35

Es un generador combinación de tres generadores congruenciales multiplicativos propuesto por P. L'Ecuyer (1987) con uso específico en microcomputadores.

$$w_{i+1} = 157\ w_i \pmod{32363}$$

$$y_{i+1} = 146\ y_i\ (\text{mod}\ 31727)$$

$$z_{i+1} = 142\ z_i (\text{mod}\ 31657)$$

de donde se obtiene:

$$x_i = (w_i + y_i + z_i - 3)\ (\text{mod}\ 32362)$$

La sucesión $\{u_i\}$ resulta entonces:

$$u_i = (x_i + 1)\ /\ 32363$$

Semillas: $w_0 = y_0 = z_0 = 1$

El período de este generador ha sido establecido como el producto de los números:

$(32362) * (31726) * (31656)\ /\ 4 = 8.12543685 * 10^{12}$

URN36, URN37 y URN38

Estos generadores fueron estudiados por Dudewicz, Karian y Marshall (E. Dudewicz, Z. Karian y R. Marshall, 1985, pp. 9-14), teniendo en cuenta su empleo en microcomputadores.

Son del tipo congruencial multiplicativo con las características que se explicitan a continuación:

URN36

$$x_{i+1} = 5^{13}\, x_i \pmod{2^{35}}$$

URN37

$$x_{i+1} = 5^{13}\, x_i \pmod{2^{31}}$$

URN38

$$x_{i+1} = 5^{13}\, x_i \pmod{2^{15}}$$

Para el generador URN36, el código propuesto simula un tamaño de palabra de 35 bits mediante dos palabras de 31 bits; siendo la semilla igual a 32767, y fue implementado en PDP-11 /23.

En el caso de URN37, los cálculos han sido orientados a la arquitectura empleada, en este caso IBM 4341, con palabras de 32 bits.

Para el generador URN38 se realizó una adaptación apropiada a procesadores de 16 bits por palabra.

URN39

Debido a Rey (1990) usa como semilla $x_0 = 2/(1+\sqrt{5})$ y la expresión general del algoritmo es:

$$x_i = [x_{i-1} + (i + x_{i-1})\ sen(i)] \pmod{1}$$

Se encuentra aún en estudio debido a problemas de exactitud, cuando la función "seno" toma valores muy grandes.

URN40

Se trata de una clase de generadores con períodos muy grandes que exceden el orden 10^7. Esta propiedad lo hace interesante por cuanto las velocidades de cálculo en los computadores son cada vez mayores, sin embargo no constituye una condición suficiente para que el generador sea de buena calidad.

Tiene un período aproximado a 4×10^{28}.

URN40, llamado Kiss, (Keep It Simple, Stupid) se debe a un trabajo previo de Marsaglia y Zaman, y utiliza tres sucesiones de números enteros.

En ellas y_i, z_i contienen valores binarios y sirven para definir finalmente la sucesión $\{x_i\}$.

Los operadores de desplazamientos de bits se designan con R y L .

$$w_i = \quad 69069\, w_{i-1} + 23606797$$

$$y_i = \quad y_i(w_i + R^{17})(w_i + L^{15})$$

$$z_i = \quad z_i(w_i + L^{18})(w_i + R^{13})$$

$$x_i = \quad w_i + y_i + z_i \pmod{2^{32}}$$

4.1.2 Un modelo computacional

Entre los últimos GNA, se destacan varios que son definidos mediante algoritmos congruenciales, módulo 2^x ($x \leq 2^{31} - 1$). Teniendo en cuenta las restricciones (cfr.3.1.1) para la representación de números muy grandes en una palabra-memoria de 32 bits, es preciso una programación específica del algoritmo para simular la descomposición de una palabra de mayor longitud en dos palabras con menores cantidades de bits.

Un modelo computacional diseñado para PC con 32 bits que evite problemas de desbordamiento, especialmente cuando el módulo de un generador supera $(2^{31}-1)$, fue programado para URN13 (Ahrens y Dieter)

El método conguencial aquí considerado es de módulo $m = 2^{32}$:

$$x_{i+1} = 663608941 \;\; x_i \pmod{2^{32}}$$

Para tal propósito simulamos una palabra de 32 bits mediante dos palabras de 31 bits cada una, para representar el producto de la constante multiplicativa fija 663608941 y el valor variable x_i.

Designamos con C , B a dichas cantidades que se descomponen en C1,C2 y B1, B2 , respectivamente, de modo que:

$B1 + B2$	$C1 + C2$

$$B = B1 * 2^{15} + B2$$

$$C = C1 * 2^{15} + C2$$

El producto $B * C$, de cada B por la constante C da lugar a las multiplicaciones de dos binomios, como se indica:

$$B * C = P + Q + R$$

$$P = (B1 * C1) * 2 * *30$$

$$Q = (B1 * C2) * 2 * *15$$

$$R = (B2 * C1) * 2 * *15 + B2 * C2$$

A partir de la fórmula precedente, se consideran separadamente a cada uno de los términos, después de efectuar los productos, con el fin de realizar los desplazamientos de bits a la izquierda y derecha en forma conveniente para formar la nueva palabra de 61 bits (aparte de un bit, para el signo).

Como en este caso el módulo es $m = 2^{32}$, es evidente que sólo interesa conservar los 32 últimos bits de la nueva palabra así formada. Precisamente ellos constituyen los números pseudoaleatorios de la sucesión a generar.

Para la partición de B, C en dos cantidades cada una, se ha tenido en cuenta que los 16 bits de más a la izquierda de cada número deben ser extraídos, como es el caso para $B2$, $C2$.

Para extraer los 15 bits de más a la derecha se han destinado las variables $B1, C1$.

Se observa que considerando el bit del signo, la suma de (16+15+1) totaliza los 32 bits de la palabra-memoria disponible.

Para ilustrar la idea precedente, en el caso de la cantidad C, la extracción de los 15 dígitos de más a la derecha, se efectúa multiplicando por una potencia de dos $(16 = 31 - 15)$ e inmediatamente después, dividiendo por la misma cantidad:

$$a \leftarrow C * 2 * *16$$

$$aa \leftarrow a \ / \ 2 * *16$$

Internamente esto implica desplazamientos de bits y el llenado con ceros en las posiciones desalojadas. En caso de haber empleado otros lenguajes de alto nivel (C, por ejemplo) tales operaciones pueden efectuarse directamente mediante instrucciones denominadas left/right shift. En este proceso también es necesario considerar cuestiones de signo, de manera que cuando las cantidades x sean negativas se ha previsto adicionarles $(2^{31} - 1) + 1)$, para permanecer en el rango $[0, 2^{31} - 1]$.

Las cantidades $B1, B2$ intervienen en el cálculo del nuevo valor que representa al nuevo número generado.

La semilla $x_0 = 524287$ ha sido almacenada en la primera componente de un arreglo de números enteros, $INTIN(1) \leftarrow x_0$.

Las otras componentes se definen de modo de realizar los desplazamientos convenientes:

$$INTIN(2) \leftarrow INTIN(1) \, / \, 2^{15}$$

$$J \leftarrow INTIN(1) * 2^{16}$$

$$INTIN(3) \leftarrow J \, / \, 2^{16}$$

$$INTIN(4) \leftarrow 663608941 \,/\, 2^{15}$$

$$INTIN(5) \leftarrow J \,/\, 2^{16}$$

Entonces, las definiciones de B, C a través de $B1$, $B2$, $C1$, $C2$ pueden escribirse en el encabezamiento del programa codificado en Fortran :

$$C1 = INTIN(4); C2 = INTIN(5)\,;\, B1 = INTIN(2)\,;\; B2 = INTIN(3)$$

Se constata que debido a la inicialización de $C1, C2$, estas cantidades permanecen constantes en el proceso, en tanto que $B1, B2$ se actualizan al salir del ciclo de iteraciones ($N\,lot$), tantas veces como esta rutina sea llamada por el programa principal ($Numalea$ muestras).

Además, puede verse como una ilustración, en las líneas tres y cinco del siguiente pseudocódigo, que la multiplicación y división se realizan, sucesivamente, por $2^{14} = 16384$ (de 61-32 bits se restan los 15 bits que se extraen, 29-15=14).

Para $k = 1, Numalea$
Para $i = 1, N\,lot$
J2 = (B2*C1 + B1*C2)*16384
Si (J2.LT.0) asignar: J2 = J2+ 2147483647 + 1
sino calcular: RSUM = J2/ 16384; J3 = B1*C1*536870912; FinSi
Si J3 $\leq$ 0 asignar: J3 = J3+ 2147483647 + 1
sino calcular: RB1C1 = J3/ 536870912; LCB2 = (C2*B2)/32768
J4 = ((LCB2 + RSUM + RB1C1 * 32768)* 16384; FinSi
Si J4 $\leq$ 0 asignar: J4+ 2147483647 + 1
sino calcular: B1 = J4/ 16384; JEN = B1 / 4; J5 = C2*B2*65536; FinSi
Si J5 $\leq$ 0 asignar: J5+ 2147483647 + 1
sino calcular: B2 = J5/65536; X(i) = FLOAT(JEN)/ 32768.0; FinSi
FinPara
Asignar: INTIN(2) = B1; INTIN(3) = B2
FinPara

4.2 PRUEBAS EMPIRICAS EN GENTSTPC

Las pruebas empíricas que forman parte de GENTSTPC son análogas a las descritas con mayor detalle estadístico en el Anexo I, bajo las denominaciones: *recolector de cupones* (TST05); *de intervalos o huecos* (TST04); *permutación* (TST07); *de póker* (TST08); *de las corridas* (TST09); *de los pares seriales* (TST10); *del producto rezagado* (TST03); y la *prueba de los máximos* (TST11), (cfr. A.2.1 hasta A.2.8).

En razón de la importancia que arrojan los resultados de la pruebas de Uniformidad (TST02) y la prueba de Kolmogorov-Smirnov (TST06), se introducen seguidamente algunos de sus fundamentos.

4.2.1 Prueba de Uniformidad

En el sistema Gentstpc, se designa con TST02 a esta prueba de frecuencia. Se trata de probar que la sucesión $\{u_i\}$ aparece uniformemente distribuida entre 0 y 1. Para ello se aplicará la prueba Chi-cuadrado, donde todos los parámetros son conocidos.

Dada una sucesión de N números reales entre 0 y 1 se procede a particionar (0,1) en k=100 intervalos: (0.00, 0.01); (0.01, 0.02); ...; (0.98, 0.99); (0.99, 1.00).

Si la distribución es realmente uniforme se espera encontrar, en promedio, en el j-ésimo intervalo, $(j = 1, 2, ..., 100)$,una misma cantidad E_j en cada uno de los 100 intervalos.

$$E_j = \frac{N}{100}$$

En realidad se encontrarán O_j observaciones que son valores en general diferentes a E_j, en esos mismos intervalos.

Luego se calculará el estadístico Chi-cuadrado que mide la discrepancia entre los valores esperados E_j y los valores observados O_j.

$$\chi^2 = \sum_{j=1}^{k} \frac{(O_j - E_j)^2}{E_j}$$

En condiciones de verdadera uniformidad en (0,1) el estadístico χ^2 tiene una distribución aproximadamente χ^2 con (n -1) = 99 grados de libertad, que notaremos χ^2_{99}

En esta etapa, fijado el nivel de significación α se compara χ^2 con $\chi^2_{n-1,\gamma}$ llamado valor crítico, incluido en la respectiva tabla estadística.

Si $\chi^2 > \chi^2_{n-1,\gamma}$,γ =1-α, se rechaza H_0 al nivel α.

4.2.2 Prueba de Kolmogorov-Smirnov

Esta prueba denominada TST06 en el paquete Gentstpc, compara la función de distribución F(x) de la distribución uniforme $F(x) = x, \quad si\ x \in [0,1]$, con la función empírica continua $S_N(x)$ de una muestra de N observaciones.

Si la muestra $U_1, U_2, ..., U_N$ es de números pseudoaleatorios, entonces la función $S_N(x)$ se define así:

$S_N(x) = (Cantidades\ U_i\ tal\ que\ U_i \leq x)\ /\ N$

La prueba K-S compara las desviaciones de F(x) con $S_N(x)$ mediante el cálculo de las siguientes medidas:

$$K^+ = \sqrt{N} max \{S_N(x) - F(x)\}$$

$$K^- = \sqrt{N} max \{F(x) - S_N(x)\}$$

$$K^* = max \left\{K^+, K^-\right\},\ x \in [0,1]$$

Un algoritmo de cálculo es:

1. Ordenamiento de la sucesión:

$$U_{(1)} \leq U_{(2)} \leq ... \leq U_{(N)}$$

2. Cálculo:

$$K^+ = \sqrt{N} max \left[\frac{i}{N} - F(U_i)\right];\ 1 \leq i \leq N$$

3. Cálculo:

$$K^- = \sqrt{N} max \left[F(U_i) - \frac{i-1}{N} \right] ; 1 \leq i \leq N$$

Si $K^* \leq K_\alpha$ (valor crítico en tablas), no se rechaza la hipótesis de que los $\{U_i\}$ son extraídos de una muestra correspondiente a una distribución uniforme.

4.2.3 Un caso particular de uniformidad: Chi cuadrado sobre Chi-cuadrado(CSCS)

En el caso que la prueba de uniformidad ya descrita en 4.2.1 se realice sobre sucesiones de 10'000 números pseudoaleatorios, la frecuencia esperada es entonces:

$$E_j = 10'000 \frac{1}{100} = 100$$

El estadístico Chi-cuadrado es:

$$\chi^2 = \sum_{j=1}^{k} \frac{(O_j - E_j)^2}{E_j} = \sum_{j=1}^{100} \frac{(O_j - 100)^2}{100}$$

En la última columna de las salidas de esta prueba programada en el sistema GENTSTPC, para cada intervalo numerado de 1 a 100, se expresa la cantidad con que cada observación O_i , y el valor esperado E_i, contribuyen con Chi-cuadrado: $(O_i - E_i)^2 / 100$.

Para mil lotes de diez mil números pseudoaleatorios, cada uno, se calcularán χ_i^2 valores.

Bajo hipótesis de uniformidad entonces los χ_i^2 ; $i = 1,1'000$; se comportarían aproximadamente como una muestra proveniente de una distribución $\chi^2_{99,\gamma}$

Esto último podría, a su vez, ser sometido a una prueba utilizando la prueba Chi- cuadrado.

Para ello se consideran 100 intervalos equiprobables para los cuales se necesita determinar los puntos porcentuales de χ^2, por ejemplo, por el método de aproximación de Cornish-Fisher.

$$I_1 = \chi^2_{99,0.01}\ ;\ I_2 = \chi^2_{99,0.02}\ ;\ \ldots\ ;\ I_{100} = \chi^2_{99,\,0.99}$$

Fijado el nivel de significación α, la probabilidad $\gamma = 1-\alpha$ (coeficiente de confianza) variará de 0.01 a 0.99 conforme a que asuma los valores 0.99, 0.98, ..., 0.01

Conocidos los anteriores puntos porcentuales, se forman los intervalos:

$$(0, I_1)\,;(I_1, I_2)\,;\ldots;(I_{99}, I_{100})$$

donde los valores esperados serán:

$E_1 = E_2 = \ldots = E_{100} = 10$ para los 1'000 valores χ^2_i, con i = 1,1'000

Calculando las 100 observaciones O_i , $i = 1,100$, en los 100 intervalos, se procede al cálculo del estadístico Chi-cuadrado Total.

$$\chi^2_T = \sum_{i=1}^{100} \frac{(O_i - 10)^2}{10}$$

el cual tiene aproximadamente una distribución χ^2_{99} , bajo hipótesis de uniformidad.

Este procedimiento puede esquematizarse en el siguiente bloque secuencial aplicable a todo generador del paquete GENTSTPC.

[**1**] Para k=1, 1'000 generar lote de 10'000 números pseudoaleatorios

1.1 Determinar las cantidades observadas O_j, de números que caen en cada uno de los 100 intervalos en que se divide (0,1)

1.2 Calcular :

$$\chi^2_k = \sum_{j=1}^{100} \frac{(O_j - 100)^2}{100}$$

finPara

[**2**] Para $\chi^2_j = 1$ con j = 1,1'000

2.1 Determinar la cantidad de χ_j^2 observados en [1] que caen en los 100 intervalos de cotas superiores, iguales a los coeficientes de Cornish-Fischer.

2.2 Calcular :

$$\chi_T^2 = \sum_{j=1}^{100} \frac{\left(\chi_j^2 - 10\right)^2}{10}$$

donde χ_T^2 tiene aproximadamente una distribución Chi-cuadrado con 99 grados de libertad, χ_{99}^2.

2.3 Calcular, para un nivel α, la probabilidad:

$$P\left[\chi_{99}^2 \leq \chi_T^2\right] = \gamma = 1 - \alpha \ (*)$$

finPara

(*) Los valores se encuentran tabulados por ejemplo en Karian y Dudewicz (K.& D., 1991, p.430).

Una notación equivalente a la de la última etapa del procedimiento anterior es $P\left[CS \leq T_{(i)}\right]$ que indica con CS el estadístico Chi-cuadrado y $T_{(i)}$ al estadístico χ_T^2 aplicado a la prueba i , índice que recorre 19 tests del paquete GENTSTPC:

1) Distribución Uniforme
2) Cupones ($D = 5$)
3) Cupones ($D = 5, FR(1000 * U)$)
4) Cupones ($D = 10$)
5) GAP encima de la media
6) GAP debajo de la media
7) GAP (intervalo 0.333-0.667)
8) Permutaciones (de 3)
9) Permutaciones (de 4)
10) Permutaciones (de 5)
11) Póker (manos de 4, 4 repartic.)
12) Póker (5,6)
13) Póker (5,4)
14) Corridas (U)
15) Corridas ($FR(10 * U)$)

16) Corridas ($FR(100 * U)$)
17) Pares seriales (3x3)
18) Pares seriales (10x10)
19) Pares seriales (20x20).

En la subrutina GAMA se calculan valores de la distribución Chi-cuadrado, con 99 grados de libertad. Llamando γ a la probabilidad que la variable aleatoria X, que tenga esa distribución, sea menor o igual que y, se adopta la siguiente notación:

G' es el valor aproximado de G = γ, para (γ =0.01(0.01)0.99). Entonces, a partir de la tabla (G,y), se calcula G', mediante interpolación, y tal que y = T(i).

Es decir: $G' = P[\, X <= T(I)\,] = P$[CHI-CUAD<= $T(I)$]; donde el estadístico T(i) = Chi-cuadrado Total es aplicado sobre 100 intervalos.

Los valores interpolados son los $y = T(i)$, sumas de 100 contribuciones a la prueba Chi-cuadrado en cada una de las 19 pruebas Chi-cuadrado sobre Chi-cuadrado(CSCS).

Cada prueba recibe 1'000 valores T(i), cada uno proveniente de 10'000 números pseudoaleatorios obtenidos por un determinado generador URN**.

4.2.4 Criterio de selección CSCS

Básicamente consiste en la realización de una cantidad de pruebas estadísticas empíricas (19) sobre mil muestras de sucesiones de diez mil números cada una, que por hipótesis se suponen aleatorias, las que producen un estadístico que se ajusta aproximadamente a una distribución Chi-cuadrado.

Luego, para cada prueba se someten esos mil valores a una nueva prueba Chi-cuadrado, en intervalos equiprobables (n =100), con el fin de obtener un nuevo estadístico que se espera tenga, bajo hipótesis de uniformidad, una distribución Chi-cuadrado, con (n-1) grados de libertad. El conjunto de estos valores para la batería de 19 pruebas, proporcionan una medida de la aleatoriedad correspondiente a grandes sucesiones de números producidos por un GNA.

La prueba TST01 del sistema Gentstpc, para la opción CSCS, tiene un particular rendimiento y efectividad, y su formato original fue dado por Dudewicz y Ralley (cfr.14), de donde su denominación CS(Chi-cuadrado), sobre CS(Chi-cuadrado).

De manera análoga se puede proceder para el caso de la distribución de Kolmogorov - Smirnov. Valores K-S provenientes de la prueba de bondad de ajuste o de la prueba de los máximos de t, pueden almacenarse y luego aplicarles nuevamente la prueba K-S.

Una facilidad de TEST01 permite además obtener separadamente, por cada prueba, un valor Chi -cuadrado, al repetir su ejecución un gran número de veces en sucesivas corridas con lotes de números pseudoaleatorios proporcionados por el generador que se esté probando.

Por otra parte, si una de las pruebas empíricas detalladas en el parágrafo A.2 (Anexo I) diera individualmente resultados pobres para una determinada sucesión, aunque en la mayoría de los otros casos tenga un buen funcionamiento, el TST01 provee nuevas mediciones que confirmarán o no los resultados esperados.

En el Cuadro N° 4.2.4.1 se listan las pruebas estadísticas de Gentstpc, una breve descripción de la función que cada uno realiza y en particular, cuando se invoca TST01, si la prueba almacena valores para luego ser empleados en las pruebas Chi-cuadrado, o Kolmogorov-Smirnov.

Una vez obtenidas las salidas para los generadores a los que se les aplica las pruebas CSCS de TST01 del paquete Gentstpc, se analizan los resultados clasificados en los siguientes casos:

a) Cantidad de resultados en que G' ≥ 0.99

b) Cantidad de resultados en que G' ≥ 0.95

c) Cantidad de resultados en que G' ≥ 0.50

Los valores *a)* indican una probabilidad muy alta; es decir, $T(i) = \chi^2_T$ es un valor extremo, y entonces la prueba i-ésima falla.

Si la probabilidad está entre 0.95 y 0.99, caso *b)*, la prueba puede dar un resultado "sospechoso", o en otros casos, un rechazo del generador analizado.

En el último caso *c)*, la probabilidad está entre 0.50 y 0.95.

Una evaluación más ajustada adopta el criterio de cuantificar el número de veces que en cada generador la probabilidad es: G'$\geqslant$0.99 y G'$\geqslant$0.90.

Según este criterio, cuando la cantidad N de veces que G'$\geqslant$0.99, es N$\geqslant$ 1, el generador es rechazado.

Si N = 0, y además, la cantidad M de veces que G'$\geqslant$ 0.95, es M $\geqslant$2, entonces, también el generador es rechazado; pero si M=1, el generador debe pasar otras pruebas, entre ellas entropía y también una prueba teórica (cfr. A.3.2, prueba espectral), a fin de corroborar o no la decisión del criterio (cfr.4.3.3, Metodología).

En el Cuadro Nº 4.2.4.2, se muestran los resultados de los generadores estudiados en los últimos años, donde se puede apreciar numerosas coincidencias con el Cuadro Nº4.2.4.3 de Karian y Dudewicz (Karian Z. y Dudewicz E., 1985). Cuando el GNA se acepta, se indica con "s", y con "n", si se lo rechaza.

Puede observarse en particular que el Generador URN13 satisface ampliamente el criterio al obtener: cuando G'$\geqslant$ 0.99, N=0; si G'$\geqslant$ 0.95, N=0; y 10 pruebas con G'$\geqslant$ 0.50.

Como resultado de la aplicación de TST01, conforme al procedimiento indicado, para ambas opciones, CS2KS2 (cfr. Manual del Usuario, Anexo II), los generadores URN01, URN02, URN12, URN13, URN14, URN22, URN28, URN30, URN31, URN35, URN36, URN37 y URN39, verifican el criterio de selección CSCS, pero no las restantes pruebas incluidas en la Metodología para la selección de un generador (cfr.4.3 y 6.2).

El Cuadro N°4.2.4.4 ilustra uno de los mejores resultados arrojados por Gentstpc para un generador, el URN13.

Por último, como una ilustración de la aplicación a otro tipo de GNA, los generadores empleando polinomios de Weyl (cfr. 3.2), al analizar los respectivos

archivos de salidas, que constan de diez millones de números pseudoleatorios cada uno, con la opción "TST01" (Chi-cuadrado sobre Chi-cuadrado, Kolmogorov-Smirnov sobre K-S), los resultados no son satisfactorios para los 1'000 lotes de 10'000 números, en ninguno de los casos.

En los Cuadros N° 4.2.4.5 y N° 4.2.4.6 se muestran los resultados para A= e y A = □, en ese orden, siendo relativamente mejores los resultados del primero de ellos.

4.3 METODOLOGIA PARA LA SELECCION DE UN GENERADOR

4.3.1 Pruebas basadas en un estimador de entropía

En el sistema GENTSTPC se proponen básicamente pruebas con propiedades empíricas para evaluar la calidad de los números pseudoaleatorios, y también algunas pruebas de carácter teórico.

Entre las pruebas estadísticas de uniformidad, en el parágrafo 4.2.4, se plantea la selección de un generador teniendo en cuenta la prueba TST01 y las opciones CSCS y CS2KS2.

A todas ellas hemos agregado un nuevo conjunto de pruebas basadas en la distribución asintótica de un estimador de entropía. Consiste en un conjunto de programas (TSTENTR) que se diseñaron como parte de las pruebas de números pseudoaleatorios incluidas en GENTSTPC. Utilizando la opción de ingreso de datos a través de un archivo (URNWN), se realizan las Pruebas de Entropía empleando los estimadores H_{mn}(Vasicek) y H*ALFA(m,n) (Dudewicz-Van der Meulen).

Una vez que TSTENTR calcula los resultados promedio de H_{mn}, entonces se obtiene la expresión del punto percentil P_m, de acuerdo a un *algoritmo* propuesto en 3.5.3.

$$P_m = 100 \times \Phi\,(\frac{F_m}{l})$$

donde l es la cantidad de muestras producidas por el generador, y F_m es la expresión del estadístico de la distribución asintótica. Para determinar P_m empleamos tablas o paquetes estadísticos-matemáticos que contengan la función de distribución normal estándar. En el mencionado *algoritmo* se proporciona un criterio mediante el cual el valor de m determina la aceptación, o rechazo, del generador estudiado.

Las pruebas de entropía son apropiadas especialmente para ser aplicadas a aquellos generadores que hayan pasado satisfactoriamente otras pruebas empíricas.

En la Figura N° 4.3.3 se observa la inclusión de entropía continua como una última prueba (TSTENTR) para determinar si un GNA es aceptado o rechazado.

4.3.2 Prueba de generadores simultáneos

En un proceso de simulación es generalmente insuficiente disponer de un solo generador de números pseudoaleatorios. En muchas aplicaciones son requeridas varias sucesiones o flujos de números aleatorios que posean propiedades de independencia entre ellos (L'Ecuyer, 1991).

Con el propósito de producir múltiples flujos se suelen calcular *semillas* que estén convenientemente espaciadas en la sucesión de números que provee un mismo generador; luego, tales semillas son usadas para determinar el punto inicial de subsucesiones que están dotadas de propiedades de independencia.

Sin embargo, esta propiedad de algunos paquetes de poseer un mismo GNA, para generar una determinada cantidad de sucesiones a partir de diferentes semillas, puede resultar limitada frente a un mayor requerimiento de flujos de números pseudoaleatorios para una aplicación.

Otros Lenguajes de Simulación en cambio son más flexibles, ya que en diferentes bloques de tareas del programa principal, permiten el acceso de distintos GNA, lo que incrementa el número de sucesiones independientes disponibles en el modelo.

Nuestro enfoque consiste en disponer de *varios* generadores que hayan superado satisfactoriamente las pruebas estadísticas empíricas, teóricas y de entropía incluidas en el sistema GENTSTPC, para luego utilizarlos como fuentes de generación de cada variable aleatoria del modelo de simulación estudiado, incluyendo en cada caso la generación de sucesiones independientes a partir de semillas que los mismos algoritmos determinan para cada flujo, según la especificación del usuario, y donde la cantidad requerida puede alcanzar, en Gentstpc, hasta diez mil números en cada flujo.

4.3.2.1 Aplicación de varios generadores en una simulación

Un aspecto que complementa la caracterización de un *buen* generador es el resultado positivo de su práctica (Hellekalec, 1998).

Con este propósito consideramos de mayor interés escoger la simulación del sistema del Cap.5 de Aplicaciones, donde se ha estudiado el comportamiento histórico del Sistema Granizo a través de muestras de datos reales en veinte temporadas de tormentas de granizo (1966-1986).

Las pruebas realizadas con *distintos* generadores *simultáneos* fueron fundamentales para ratificar las ventajas de unos y la ineficiencia de otros, al ser empleados para simular el comportamiento de las distintas variables.

En esta aplicación, las variables principales consideradas en el modelo son: *cantidad de tormentas*; *tipos de tormentas* por temporada; *cantidad* de tormentas por cada una de las *localidades* en la región analizada; y el *daño económico* total en la temporada para dos tipos de frutales.

En la Figura Nº 4.3.2.1, se resume el proceso de simulación con generadores simultáneos. Intervienen seis GNA cuya eficiencia fue comprobada con anterioridad; seis GNA no eficientes; y los RN_i(i =1,6) del único generador propio del lenguaje empleado (GPSS World), a modo de comparación.

Los GNA eficientes seleccionados son: URN13 para generar "tipos de tormentas"; URN12 para la "cantidad de tomentas"; URN22 para "localidades" afectadas; y URN01, URN35 y URN39, para generar tres cantidades que representan "superficies" afectadas.

En la selección de los GNA ineficientes, dichas variables aleatorias fueron generadas, respectivamente por URN40, URN08, URN34, URN05, URN06, y URN21.

4.3.2.2 Resultados

En el análisis estadístico de los resultados, consideramos al sistema en estado estacionario (steady-state), y con el propósito de reducir la correlación de las sucesiones de variables aleatorias generadas, aplicamos la técnica de una "corrida larga", para la recolección de un gran número de observaciones, particionada en subsucesiones.

Para estimar los parámetros (Θ) que representan las distintas medidas de rendimiento del sistema en régimen estacionario, tenemos en cuenta la definición (Bank y otros, 1996) de Θ:

$$\Theta = \lim_{n \to \infty} \frac{1}{n} \sum_{i=1,n} X_i$$

A este valor Θ se espera que converjan las *medias* de las muestras de las sucesiones $\{X_i\}$ producidas por diferentes números aleatorios.

En nuestra aplicación fijamos la longitud de la "corrida larga", en $n = 2'000'000$.

Habiendo elegido la técnica de las "medias de lotes", se fija una partición de la cantidad n de corridas en $K = 20$ sucesiones o *lotes* del mismo tamaño L =100'000. Luego, para el cálculo de la *media* de cada lote L_j, se tiene:

$$\overline{X_j} = \frac{1}{n} \sum_{i=1,L} X_i \ , \ j = 1, ..., K$$

donde $\overline{X}$ representará la *media total* de las medias $\overline{X_j}$.

Simultáneamente, se calculan los correspondientes intervalos de confianza (IC) que incluyen las *medias*, para un coeficiente de confianza dado, el cual representa la probabilidad de incluir en los IC a los verdaderos valores de las *medias* Θ_i, desconocidas, de cada medida de rendimiento.

La elección de la cantidad de lotes K coincide con los valores propuestos por Schmeiser (Schmeiser B., 1982), ya que al respecto, no hay un criterio único aceptado.

Las etapas de la simulación desarrolladas en la aplicación de la técnica de las "medias de lotes", de la ya citada Figura N° 4.3.2.1, abarcan dos ciclos, según intervengan los GNA eficientes o ineficientes, adoptando la notación $GNA1_i$, $GNA2_i$, ... ,$GNA6_i$ para los seis generadores. Los archivos indicados con 100'000 son las sucesiones de números pseudoaleatorios disponibles en cada uno de los L_k lotes
(k =1,20) de la técnica aplicada.

Las variables, cuyas medidas de rendimiento interesan más en este modelo, son: la *cantidad media*, Θ_1, de tormentas de los *tipos* "1" a "4 ", por temporada; la *cantidad media*, Θ_2, de tormentas por *localidad* (de 1 a 21); y el daño económico *promedio*, Θ_3, para manzanas y peras en la región.

La primera, Θ_1, aparece desagregada en cuatro valores, según los tipos de tormenta, de donde es posible deducir las cantidades porcentuales de tormentas por cada tipo, Figuras N°4.3.2.2.1 y N°4.3.2.2.2; Θ_2 se calcula para cada una de las localidades; y las estimaciones del parámetro Θ_3, se presentan en la Figura N°4.3.2.2.3

Los resultados de las Figuras N° 4.3.2.2.1 fueron obtenidos con la intervención de GNA eficientes, mientras que los de la Figura N° 4.3.2.2.2 corresponden a los GNA no eficientes.

Del análisis estadístico efectuado, al estimar las medias Θ_1, con un coeficiente de confianza del 95%, podemos concluir que los resultados aportados por los GNA eficientes, concuerdan satisfactoriamente con el comportamiento del sistema real.

En efecto, 45% de las tormentas, en promedio, son del tipo uno, 38% del tipo dos, y para los tipos tres y cuatro alcanzan el 14% y 3%, respectivamente; estos valores coinciden exactamente, en porcentaje, con los arrojados por los GNA (RNi) del lenguaje de simulación utilizado independientemente con fines de comparación. En cambio, se aprecian diferencias en todos los porcentajes de tipos de tormentas para los GNA ineficientes (Figura Nº 4.3.2.2.2).

Un análisis estadístico similar para estimar Θ_3, permitió observar serias discrepancias entre los valores de las perdidas económicas, medidos en unidades monetarias, para los daños de las tormentas de granizo en cultivos de manzanas y peras, cuando se emplean generadores eficientes e ineficientes, como se muestra en la Figura Nº4.3.2.2.3.

Los resultados correspondientes a las medias de tormentas según localidades,Θ_2, en la región analizada, no permiten marcar diferencias considerables entre los resultados arrojados por los diversos generadores. La menor probabilidad de correlación en dicha variable aleatoria puede inferirse que es causada por el uso de una cantidad relativamente inferior de números aleatorios en las sucesiones disponibles, ya que cada sorteo de una localidad debe distribuirse entre 21 localidades posibles.

A modo de conclusión, puede afirmarse que los resultados estadísticos, confirman el funcionamiento satisfactorio de los GNA eficientes seleccionados, interactuando simultáneamente en el proceso de simulación de este modelo.

4.3.3 Metodología

Teniendo presente por una parte, que "los diferentes tipos de pruebas sirven para detectar distintos tipos de deficiencias, y lo mejor es disponer de una batería que sea lo más diversificada" (cfr. L'Ecuyer P., Simard R. y Wegenkittl S., *Sparse Serial Tests of Uniformity for Random Number Generators*, SIAM Journal on Scientific Computing, 24, 2 (2002), 652-668. *http://www.iro.umontreal.ca/~lecuyer/myftp/papers/serial.pdf*); y por otra parte, el contexto en el que se plantea la generación y la verificación de los números pseudoaleatorios en microcomputadores, se ha arribado a una metodología posible para la selección de un generador, la cual se esquematiza en la Figura N°4.3.3.

En la primera parte se desarrolla la prueba de uniformidad TST01 (cfr. 4.2.4) del sistema GENTSTPC, con la opción Chi-cuadrado sobre Chi-cuadrado y Kolmogorov-Smirnov sobre K-S, CS2KS2, conforme al respectivo algoritmo.

La segunda parte hace intervenir la prueba de Gentstpc sobre Entropía continua (rutinas TSTENTR).

La Metodología elaborada para la selección de una GNA, (cfr. Figura N° 4.3.3) puede expresarse a través de estas etapas:

E1) Introducción del Generador

En Gentstpc se dispone de un programa-menú que inicialmente presenta dos opciones:

• *Generación de números pseudoaleatorios sistema Gentstpc. Permite elegir los GNA del sistema, es decir URN01, URN02,..., URN40.*

• *Generación de números pseudoaleatorios por otros métodos. La programación de los respectivos algoritmos es externa a Gentstpc; sólo se requiere un archivo tipo texto con un formato y nombres determinados, como se indica en las ventanas de ayuda del paquete.*

E2) Pruebas estadísticas empíricas (TST01, opción CS2KS2)

Después que el GNA es ingresado, el sistema crea varios archivos para realizar una batería de pruebas. Se seleccionará la opción *Tests Empíricos* y luego TST01, que ejecuta la prueba Chi-cuadrado sobre los resultados Chi-cuadrado aplicados a las 19 pruebas (cfr. 4.2.3). La siguiente opción, CS2KS2, aplica reiteradamente

las pruebas Chi-cuadrado(CS2) y Kolmogorov-Smirnov(KS2), según se detalla en 4.2.4.

Los resultados se muestran en un cuadro de tres columnas (cfr.URN13, Cuadro N° 4.2.4.4): la primera indica los nombres de las 19 pruebas y las diversas cantidades G'; la segunda, los valores del estadístico total T(i); y la tercera, los valores de la probabilidad G'.

NOTA: En *4.3.3* hemos definido a G' como el valor aproximado de la probabilidad que una variable aleatoria X tenga la distribución Chi-cuadrado con 99 g.l. Dicha probabilidad debe ser menor o igual que T(i), estadístico que representa el Chi-cuadrado total, es decir la suma de las 100 contribuciones de cada una de las pruebas i, ($i = 1, 2, ..., 19$, es decir distribución uniforme, cupones,..., pares seriales, respectivamente); y que CSCS consiste en aplicar reiteradamente Chi-cuadrado a los valores obtenidos de la aplicación del mismo estadístico en cada prueba i, sobre 1'000 muestras de $\{u_1, u_2, \ldots, u_{10000}\}$ números cada una, siempre bajo hipótesis de aleatoriedad.

E3) Verificación de los resultados de la Etapa 2

Consiste en la aplicación de un criterio de selección referido a G'. Se designa N a la cantidad de veces que G'$\geqslant$ 0.99; M, al número de veces que 0.95 $\leqslant$G' $<$ 0.99:

a) El GNA no verifica las pruebas si $\mathbf{N \geqslant 1}$*; entonces, es ineficiente. Fin de las pruebas.*

b) El GNA no satisface las pruebas si $\mathbf{N = 0}$ **y** $\mathbf{M \geqslant 2}$*; alto.*

c) El GNA es sospechoso si $\mathbf{N = 0}$ **y** $\mathbf{M = 1}$ *; entonces debe someterse a otras pruebas.*

Según la metodología propuesta, un generador que supere la prueba TST01, con la opción CS2KS2, y luego verifique la prueba de Entropía, entonces será un generador satisfactorio; pero si no pasa la primera prueba y sólo verifica entropía, resultaría sospechoso, puesto que "la uniformidad en entropía, considerada aisladamente, no garantiza buenas propiedades de aleatoriedad" (Bernhofen, Dudewicz, Levendovszky y Van der Meulen, p.68).

Dos ejemplos opuestos de aplicación de la metodología son el generador URN10, que no satisface las pruebas empíricas, de entropía, ni la espectral; y URN13, en cambio, que satisface todas las pruebas.

Un generador congruencial lineal, en particular, puede ser sometido además, a las pruebas teóricas, en particular la prueba TSTM4 de las distancias interplanares o espectral (cfr. A.3.2). En el Anexo II (Manual del Usuario de Gentstpc) se indica el procedimiento de ingreso al sistema de los parámetros del generador en estudio: constante multiplicativa (a) y módulo (m).

Una vez elegido un generador eficiente del sistema Gentstpc, se puede optar por almacenar la cantidad de números que se desee usar en una aplicación mediante la opción que crea un archivo de datos tipo texto. Todas las otras opciones posibles, son descritas en el citado Manual del sistema.

Cuadro N° 4.1.1 Generadores

Nombre	Nombre Genérico	Autor	Referencias
URN01	LLRANDOM	Learmouth y Lewis	Rep. Naval-Post Gr. School(1973) Lenguaje:Assembler
URN02		Marsaglia y Bray	CACM V.11(1968) Lenguaje:Fortran
URN03	SUPER-DUPER	Marsaglia, Paul y Ananthanarayanan	McGill School of C.S.
URN04	LLRANDOM con shu- ing		
URN05	GFSR	Lewis y Payne	JACM Vol.20(1973) Lenguaje:Fortran
URN06		Lurie y Mason	Com.Stat.-Vol.2(73) Lenguaje:Fortran
URN07	KERAND	Keane	Vers. RANDU-1969 Lenguaje:Assembler
URN08	RANDU	IBM	SSP–1970 Lenguaje:Fortran
URN09	RANDU	Nie, Bent y Hull	SPSS-1973 Lenguaje:Fortran
URN10	OMNITAB II	Kruskal	CACM v.12(1969) Lenguaje:Fortran
URN11		Coveyou y McPherson	JACM v.14(1967) Lenguaje:Fortran
URN12			V.Doble prec.URN11
URN13		Ahrens y Dieter	Graz Inst.Math.St.(74) Lenguaje:Assembler
URN14		Zarling	Lenguaje:Fortran
URN15-20		Hoaglin	NS-340 Harvard St.Dp. Lenguaje:Assembler

Cuadro N° 4.1.1 Generadores (continuación)

Nombre	Nombre Genérico	Autor	Referencias
URN21		Swain C. y Swain M.	Lenguaje:Fortran
URN22	MTH$RANDOM	GPSS-VX	Lenguaje:Fortran
URN23		Brody T.A.	Lenguaje:Assembler
URN24		Thesen A.	Lenguaje:Assembler
URN25		Gason J.	Lenguaje:Assembler
URN26		Mathews y Karian	Lenguaje:Assembler
URN27	GPSS/H	Whittlessey	Lenguaje:Fortran
URN28		Lewis y Lehmer	Lenguaje:Fortran
URN29		Schmidt D.	Lenguaje:Fortran
URN30	SIMSCRIPT II.5	Lehmer, Marse y Roberts	Lenguaje:Fortran
URN31-33		Miyazaki	Lenguaje:Fortran
URN34		Sezgin, Karian y Dudewicz	Lenguaje:Fortran
URN35		L 'Ecuyer P.	Lenguaje:Fortran
URN36-38		Dudewicz, Karian y Marshall	Lenguaje:Fortran
URN39		Rey W.J.J.	Lenguaje:Fortran
URN40		Marsaglia y Zaman	Lenguaje:Fortran

Cuadro N° 4.2.4.1 Descripción pruebas empíricas

Nombre y Descripción de las Pruebas	Genera val. CSCS	Genera val. KSKS
TST01 realiza Chi-Cuadrado sobre Chi2-Cuadrado y K-S sobre K-S		
TST02 Distribución Uniforme(Chi-Cuadrado simple)	sí	
TST03 Test del producto rezagado		
TST04 Gap	sí	
TST05 Recolector de cupones	sí	
TST06 Bondad de ajuste de K-S	sí	sí
TST07 Permutación	sí	
TST08 Póker	sí	
TST09 Corridas	sí	
TST10 Pares Series	sí	
TST11 Máximo de t		sí
Tests Especiales para Generadores Lineales:		
TSTM1 Correlación Serial		
TSTM2 Potencia y Período		
TSTM3 Factorización en Factores Primos		
TSTM4 Test Espectral		
TSTENTR Tests de entropía continua		

Cuadro N° 4.2.4.2 Selección de generadores (I)

Generador	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**
N $\geqslant$ 0.99	0	0	2		4	1		3	3	19	2	0
N $\geqslant$ 0.95	0	0	3		6	4		5	5	19	2	1
N $\geqslant$0.50	8	13	10		13	11		11	11	19	11	11
Pasa?(s/n)	s	s	n		n	n		n	n	n	n	s
Generador	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
N $\geqslant$ 0.99	0	0						18	13	0		
N $\geqslant$ 0.95	0	0						18	13	0		
N $\geqslant$ 0.50	10	11						19	18	7		
Pasa?(s/n)	s	s	†	†	†	†	†	n	n	s	†	†
Generador	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**
N $\geqslant$ 0.99				0		0	0	1	6	6	0	0
N $\geqslant$ 0.95				0		1	1	4	8	6	1	0
N $\geqslant$ 0.50				8		9	10	9	16	8	12	13
Pasa?(s/n)	†	†	†	s	†	s	s	n	n	n	s	s
Generador	**37**	**38**	**39**	**40**								
N $\geqslant$ 0.99	0	19	0	19								
N $\geqslant$ 0.95	1	19	0	19								
N $\geqslant$ 0.50	11	19	9	19								
Pasa?(s/n)	s	n	s	n								

†: Generador no verificado en Gentstpc (cfr.Anexo I, Introducción)

Cuadro N° 4.2.4.3 Selección de generadores (II)

Generador	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**
N $\geqslant$ 0.99	0	0	0	1	3	2	3	3	3	19	1	0
N $\geqslant$ 0.95	0	0	1	1	6	4	5	4	4	19	3	0
Pasa?(s/n)	s†	s	s	n	n	n	n	n	n	n	n	s
Generador	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
N $\geqslant$ 0.99	0	0	0	0	1	1	0	0	13	0	1	8
N $\geqslant$ 0.95	1	0	1	2	1	1	2	2	13	1	3	10
Pasa?(s/n)	s	s	s	n	n	n	n	n	n	s	n	n
Generador	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	
N $\geqslant$ 0.99	1	19	11	0	19	0	4	0	0	5	0	
N $\geqslant$ 0.95	3	19	14	0	19	0	5	1	3	5	1	
Pasa?(s/n)	n	n	n	s†	n	s	n	s†	n	n	s	

†: Generador que falla en la prueba espectral

Cuadro N° 4.2.4.4 Aplicación TST01 al generador URN13

Test TST01	T(I)	P[ChiCuad] $\leqslant$ T(I)
DISTR.UNIF.(TST02)	92.40	0.33
CUPONES D=5(TST04)	97.20	0.20
CUPONES FR(100R)	95.40	0.41
CUPONES D=10	98.40	0.50
GAP ARRIBA MEDIA(TST05)	82.60	0.11
GAP BAJO MEDIA	112.60	0.83
GAP (0.33,0.667)	102.40	0.61
PERMUTACIONES 3S(TST07)	91.40	0.30
PERMUTACIONES 4S	99.40	0.53
PERMUTACIONES 5S	113.40	0.84
POKER (4,4) (TST08)	107.80	0.74
POKER (5,4)	90.40	0.28
CORRIDAS R(TST09)	102.60	0.61
CORRIDAS FR(10R)	93.80	0.37
CORRIDAS FR(100R)	119.00	0.92
PARES SERIALES 3X3(TST10)	77.00	0.04
PARES SERIALES 10X10	103.00	0.62
PARES SERIALES 20X20	117.00	0.89
CANT. DE G'= P[Chi-Cuad $\leq$ T(I)]	$\geqslant$ 0.99	0
CANT. DE G'= P[Chi-Cuad $\leq$ T(I)]	$\geqslant$ 0.95	0
CANT. DE G'= P[Chi-Cuad $\leq$ T(I)]	$\geqslant$ 0.50	10

Cuadro N° 4.2.4.5 Prueba generador Urnwn (e)

GENERADOR URNWN : AWEYL (e)		
PRUEBA	T(I)	P[$\chi^2 \leq$T (I)]
DISTR.UNIF.(TST02)	121.40	0.94
CUPONES D=5(TST04)	95.20	0.41
CUPONES FR(100R)	103.40	0.63
CUPONES D=10	90.80	0.29
GAP ARRIBA MEDIA(TST05)	99.20	0.52
GAP BAJO MEDIA	88.40	0.23
GAP (0.33,0.667)	95.00	0.40
PERMUTACIONES 3S(TST07)	116.40	0.89
PERMUTACIONES 4S	99.00	0.51
PERMUTACIONES 5S	106.00	0.70
POKER (4,4) (TST08)	88.40	0.23
POKER (5,6)	89.40	0.25
POKER (5,4)	108.00	0.74
CORRIDAS R(TST09)	87.20	0.20
CORRIDAS FR(10R)	109.00	0.76
CORRIDAS FR(100R)	115.80	0.88
PARES SERIALES 3X3(TST10)	111.20	0.81
PARES SERIALES 10X10	112.80	0.83
PARES SERIALES 20X20	140.40	1.00
CANT. DE G'= P[CHI-CUAD $\leqslant$ T(I)] $\geqslant$ 0.99	1	
CANT. DE G'= P[CHI-CUAD $\leqslant$ T(I)] $\geqslant$ 0.95	1	
CANT. DE G'= P[CHI-CUAD $\leqslant$ T(I)] $\geqslant$ 0.50	12	

Cuadro N° 4.2.4.6 Prueba generador Urnwn (□)

GENERADOR URNWN : AWEYL (□)		
PRUEBA	T(I)	P[$\chi^2 \leq$T (I)]
DISTR.UNIF.(TST02)	142.80	1.00
CUPONES D=5(TST04)	67.80	0.00
CUPONES FR(100R)	120.60	0.93
CUPONES D=10	88.00	0.22
GAP ARRIBA MEDIA(TST05)	101.40	0.58
GAP BAJO MEDIA	94.80	0.39
GAP (0.33,0.667)	114.00	0.85
PERMUTACIONES 3S(TST07)	83.60	0.13
PERMUTACIONES 4S	100.00	0.54
PERMUTACIONES 5S	99.80	0.54
POKER (4,4) (TST08)	107.60	0.73
POKER (5,6)	95.40	0.41
POKER (5,4)	130.00	0.98
CORRIDAS R(TST09)	88.00	0.22
CORRIDAS FR(10R)	94.00	0.37
CORRIDAS FR(100R)	84.40	0.14
PARES SERIALES 3X3(TST10)	124.80	0.96
PARES SERIALES 10X10	102.60	0.61
PARES SERIALES 20X20	106.80	0.72
CANT. DE G'= P[CHI-CUAD $\leqslant$ T(I)] $\geqslant$ 0.99	1	
CANT. DE G'= P[CHI-CUAD $\leqslant$ T(I)] $\geqslant$ 0.95	3	
CANT. DE G'= P[CHI-CUAD $\leqslant$ T(I)] $\geqslant$ 0.50	11	

Figura N° 4.3.2.1 Proceso de simulación con generadores simultáneos

Figura N° 4.3.2.2.1 Tipos de tormentas (generadores eficientes)

Figura N° 4.3.2.2.2 Tipos de tormentas (generadores ineficientes)

Figura N° 4.3.2.2.3 Daños económicos: generadores eficientes e ineficientes

Figura N° 4.3.3 Metodología para la selección de un generador

5. APLICACIONES

Se presentan suscintamente dos clases de aplicaciones de números pseudoaleatorios, generados conforme a los conceptos teóricos y a los algoritmos ya desarrollados en los capítulos precedentes.

La primera, trata sobre el análisis realizado en paquetes matemáticos y estadísticos para evaluar la calidad de sus propios números pseudoaleatorios. La segunda, describe tres aplicaciones a modelos de simulación en diferentes campos. Naturalmente, esta selección podría ampliarse a muchos casos existentes en otras disciplinas científicas.

5.1 GENERADORES EN LENGUAJES Y PAQUETES

En primer término, haremos una referencia sintética a los lenguajes de simulación, donde el empleo de los GNA es más frecuente y específico que otros lenguajes de propósitos generales, o en los paquetes de programas numéricos y simbólicos.

El lenguaje GPSS (General Purpose Systems Simulations) de IBM, concebido por Geoffrey Gordon y R. Efron, en 1961, adopta el enfoque de "procesos". Además de las primeras versiones originales para IBM conocidas como GPSS II, GPSS/360 y GPSS V, se conocían las versiones GPSS/1100, de UNIVAC; GPSK de Honeywell y GPSS-SIAS de Siemens, entre otras.

En los últimos años aparecieron las versiones para microcomputadores que se describen sintéticamente más adelante, en el Cuadro N° 5.1.1, bajo las denominaciones GPSS/H, GPPS-PC, GPPS World, etc.

Entre los lenguajes orientados a "eventos", SIMSCRIPT fue desarrollado por H. M. Markowitz y B. Dimsdale en la Rand Corporation, en 1960. La última versión es SIMSCRIPT II.5 (CACI).

MODSIM II (CACI), puede ser considerado entre los primeros lenguajes orientado-objeto que aparecieron en el mercado. Algunas de sus características específicas son su modularidad, orientación a objetos, estructura en bloques y capacidad para la simulación de eventos discretos.

Para Taha (cfr.72), los dos lenguajes más destacados de programación de eventos son SIMSCRIPT y GASP IV; al que se agrega SLAM, variante de GASP IV;

y SLAM II (A. Pritzker) que representa al sistema por medio de entidades que fluyen a través de una red formada por nodos y actividades.

En Europa los lenguajes de "examen de actividades" son más divulgados. Un ejemplo es el ECSL (Extended Control and Simulation Lenguaje). Esta versión fue programada en la Universidad de Birmingham y fue escrita en Fortran.

Otro lenguaje orientado a procesos que se puede mencionar es SIMAN desarrollado por B. Zeigler y T. Oren. Tanto SLAM como SIMAN (Systems Modeling Corporation) permiten realizar simulación continua.

Finalmente, mencionaremos una clase de lenguajes, que están basados en redes como el Q-GERT (escrito en Fortran), el nuevo SLAM; y SIMNET, para modelos complejos, el cual posee recursos más poderosos que los otros.

En el citado Cuadro N° 5.1.1 se incluyen otros paquetes disponibles para PC, enfatizando en los productos destinados a la simulación de eventos discretos. Parte de la lista fue extraída de la encuesta realizada por James J. Swain (cfr.71) que en 2003 incluyó 48 productos del mercado, omitiendo aquéllos cuya principal capacidad es la simulación continua (p. ej. sistemas de ecuaciones diferenciales observados en sistemas físicos, o para entrenamiento, como vehículos simuladores).

La primera columna del cuadro mencionado muestra los nombres de los productos; la segunda, los requerimientos en memoria de la PC y los sistemas operativos que utilizan; y la tercera, las principales características, como tipos de usos y capacidades de producir animación.

Con relación a los paquetes de programas, y con el fin de verificar las propiedades de aleatoriedad de los generadores incorporados a cada sistema en particular, se eligieron tres paquetes comerciales matemáticos-estadísticos conocidos (MAPLE V, MATLAB, GAUSS); dos lenguajes de alto nivel y de propósitos generales (NDP Fortran v. 4.2.1 en DOS, y Microsoft FORTRAN Visual Workbench v.1.00, para Windows); y una Planilla de Cálculo (Microsoft Office Excel).

De las pruebas a que fue sometido Excel en 2001, L'Ecuyer (cfr.38) concluye en que debería ser excluido para ser usado en "aplicaciones serias". Sin embargo, la versión 2003 fue verificada satisfactoriamente por GENTSTPC, por lo que consideramos que su nuevo generador puede recomendarse entre los GNA eficientes. (cfr.Wichman B. A. y Hill I.D., y *http://support.microsoft.com/default.aspx?scid=kb;en-us;828795&Product=OFF2003*)

En los Cuadros N° 5.1.2 y N° 5.1.3 se muestran los resultados obtenidos al aplicar sucesivamente las pruebas del sistema GENTSTPC: TST01 (CS2KS2) y TSTENTR, con las alternativas que se detallaron anteriormente en la metodología del Cuadro N° 4.3.3, para aceptación o rechazo de un GNA. En consecuencia, se rechaza el generador del sistema Gauss(URN19), que tampoco supera la prueba teórica espectral; mientras que el generador del lenguaje Microsoft FORTRAN, resulta sospechoso para las pruebas de TST01(CS2KS2). En este último caso, en que el fabricante no provee la expresión matemática del generador, no es posible aplicar ninguna prueba teórica.

En el caso particular del paquete estadístico GAUSS, las observaciones se basan en investigaciones sobre series de tiempo estacionarias empleadas en Macroeconomía, y realizadas en el ámbito académico (Chen B., McCoskey S.K. y Kao Ch., Syracuse University, 1999), a fin de estudiar las propiedades de muestras finitas en una regresión cointegrada de un panel de datos, correspondientes a un estimador LSDV y a una distribución **t**.

Mediante Montecarlo, se analizan los sesgos y las divergencias de ambos para arribar a conclusiones sobre las distribuciones asintóticas del estimador LSDV y la distribución **t**, particularmente útiles en econometría aplicada.

Los números aleatorios utilizados son provistos por generadores congruenciales lineales: el generador del paquete GAUSS y el generador URN22 del paquete GENTSTPC.

En los Cuadros N° 5.1.4 y 5.1.4 (continuación) se muestran los resultados comparativos para ambos generadores, indicados en las columnas GAUSS y URN22.

Los resultados de Montecarlo expresan: medias sesgadas $(\widehat{\beta}-\beta)$ del estimador LSDV, medias corregidas $(\widehat{\beta}^{*}-\beta)$ y desviación estándar (valor entre paréntesis); medias y medias corregidas de la distribución **t** y las respectivas desviaciones estándar.

Se puede constatar que para casi todos los casos, las variaciones de los resultados para ambos métodos de generación, no exceden los dos centésimos, con excepción de $\mathbf{t}_{\beta}$, cuando $\square = \pm 0.4$ y $\Theta = 0.0$; y para $\widehat{\mathbf{t}}^{*}$, cuando $\square = 0.4$ y $\Theta = 0.0$

Como tales números representan promedios de muestras, entonces las tres excepciones son significativas y conducen a aceptar al generador URN22 como superior al del paquete GAUSS.

Por otra parte, la mejor calidad de URN22 con respecto al generador empleado por el paquete GAUSS (URN19) se ha verificado también aplicando el "TEST01", para 10 millones de números generados, en la opción CS2KS2 de GENTSTPC, del parágrafo sobre pruebas empíricas en Cap.4., como lo muestra el Cuadro N° 4. 2.4.2

Cuadro N° 5.1.1 Paquetes para simulación en PC

Producto	Requ. memoria - S.O.	Principales características
ALPHA	64MB-Windows, SUN	Reing. proc.comerc., manuf., control; basado
		en redes Petri, prop. grales., eventos disc. (†)
Arena	64MB-Windows	Manufact.,proc.comerc.,salud,milit. (†)
@RISK	16MB-Windows 9x,	Anál.costos;simul. Montecarlo, probl. riesgo
	NT, MAC	
AutoMod	32MB-Windows 9x,	Progr.producc. y distrib., modelos p/materiales
	NT	industria autom., aeroesp., etc. (†)
COMNET	64MB-Windows 9x, NT	Comunic.datos, telecom., redes
	UNIX,HP/UX, SGI	
GPSS/H	8MB-Win,OS2, Solaris	Leng. de propósitos generales (†)
GPSS/PC	640KB-DOS	Aplicaciones en general (†)
micro-GPSS	410KB-DOS,Win9x,	Eventos discretos, fines educ.,comercio,ing. (†)
	NT	
GPSS	32MB-Windows	Eventos discretos,redes colas,manufac.,telecom.
World		intereactivo(†)
MODSIM	32MB-WIN, Sun,	Leng. orient. objeto, probl. transp.,trafico aéreo,
III	Solaris, HP/UX, SGI	logíst.,milit.,telecomunicaciones.
PASION	64MB-Windows	Eventos discretos,colas,mod.continuos,ec.dif.
		ord. Sist.basado en Borland Delphi (†)
SansGUI	64MB-Windows	Modelos educ., medio ambiente, inv.cient.,
		orient. objetos (†)
SimGauss	16MB-DOS,Windows	Interact. con Gauss,ing,salud,finanzas
STELLA	32MB-Windows,	Constr.modelos mentales y simulación,
	Macintosh	aplic. salud, medio ambiente, fines educ. (†)
VisSim	64MB-Windows,	Sist.dinámicos no lineales, control, plantas
	Solaris	virtuales,capac. adm. exper., optim.redes (†)

†: Con animación.

Cuadro N° 5.1.2 Aplicación TST01 (CS2KS2)

Nombre Soft	Microsoft FORTRAN V.W. v.1.0	NDP Fortran v.4.2.1	MAPLE V R4	MATLAB v.4.0	GAUSS v.3	MS Office EXCEL v.2003
N $\geqslant 0.99$	1	0	0	0	0	0
N $\geqslant 0.95$	1	0	1	1	2	1
N $\geqslant 0.50$	7	10	7	9	8	11
Pasa (s/n)	n	s	s	s	n	s

Cuadro N° 5.1.3 Prueba de entropía

Nombre Soft	MS Fortran V.W.v1.0	NDP Fortran v.4.2.1	MAPLE V R4	MATLAB v.4.0	GAUSS v.3	MS Office EXCEL v.2003
$P_m = 100 \times \Phi(\cdot)$ $5 \leq P_m \leq 50$	satisface	satisface	satisface	satisface	no satisface	satisface

Cuadro Nº 5.1.4 Prueba sistema Gauss

Valores: $(\hat{\beta} - \beta)$

	Gauss	Urn22	Gauss	Urn22	Gauss	Urn22	Gauss	Urn22	Gauss	Urn22
$\theta \mid \square$	-0.8		-0.4		0.0		0.4		0.8	
-0.8	-.0147	-.0148	-.0097	-.0097	-.0016	-.0014	.0145	-.0145	.0572	.0571
	(.0056)	(.0056)	(.0059)	(.0059)	(.0065)	(.0066)	(.0079)	(.0079)	(.0119)	(.0119)
-0.4	-.0238	-.0239	-.0177	-.0179	-.0095	-.0096	.0024	.0024	.0217	.0216
	(.0071)	(.0071)	(.0073)	(.0074)	(.0077)	(.0077)	(.0079)	(.0079)	(.0079)	(.0078)
0.0	-.0395	-.0395	-.0291	-.0291	-.0189	-.0187	-.0083	-.0083	.0021	.0021
	(.0096)	(.0096)	(.0092)	(.0092)	(.0086)	(.0085)	(.0074)	(.0074)	(.0056)	(.0056)
0.4	-.0658	-.0659	-.0411	-.0411	-.0256	-.0255	-.0148	-.0148	-.0069	-.0070
	(.0135)	(.0135)	(.0109)	(.0109)	(.0088)	(.0087)	(.0067)	(.0067)	(.0046)	(.0045)
0.8	-.1039	-.1038	-.0488	-.0488	-.0284	-.0283	-.0176	-.0175	-.0109	-.0110
	(.0189)	(.0188)	(.0118)	(.0117)	(.0083)	(.0082)	(.0060)	(.0059)	(.0039)	(.0039)

Valores: $(\hat{\beta}^* - \beta)$

	Gauss	Urn22	Gauss	Urn22	Gauss	Urn22	Gauss	Urn22	Gauss	Urn22
$\theta \mid \square$	-0.8		-0.4		0.0		0.4		0.8	
-0.8	.0098	.0098	.0147	.0136	.0191	.0193	.0279	.0279	.0390	.0389
	(.0055)	(.0055)	(.0060)	(.0060)	(.0068)	(.0069)	(.0084)	(.0084)	(.0119)	(.0118)
-0.4	.0037	.0035	.0049	.0047	.0059	.0057	-.0059	-.0059	.0033	.0032
	(.0068)	(.0069)	(.0073)	(.0074)	(.0078)	(.0078)	(.0081)	(.0081)	(.0078)	(.0077)
0.0	-.0071	-.0071	-.0086	-.0086	-.0103	-.0101	-.0117	-.0117	-.0132	-.0132
	(.0091)	(.0091)	(.0091)	(.0091)	(.0086)	(.0086)	(.0076)	(.0076)	(.0058)	(.0057)
0.4	-.0239	-.0240	-.0254	-.0253	-.0243	-.0241	-.0223	-.0223	-.0205	-.0205
	(.0129)	(.0129)	(.0109)	(.0109)	(.0089)	(.0089)	(.0069)	(.0069)	(.0048)	(.0048)
0.8	-.0528	-.0526	-.0425	-.0424	-.0336	-.0335	-.0275	-.0275	-.0234	-.0234
	(.0185)	(.0185)	(.0120)	(.0119)	(.0086)	(.0086)	(.0064)	(.0063)	(.0043)	(.0043)

Cuadro Nº 5.1.4 Prueba sistema Gauss (continuación)

Valores: t_{β}

	Gauss	Urn22	Gauss	Urn22	Gauss	Urn22	Gauss	Urn22	Gauss	Urn22
θ|□	-0.8		-0.4		0.0		0.4		0.8	
-0.8	-2.898	-2.907	-1.758	-1.767	-0.256	-0.233	2.047	2044	6.255	6.252
	(1.050)	(1.046)	(1.067)	(1.065)	(1.079)	(1.082)	(1.097)	(1.096)	(1.089)	1.083
-0.4	-3.952	-3.976	-2.824	-2.850	-1.434	-1.453	.0332	0.344	2.801	2.796
	(1.085)	(1.092)	(1.129)	(1.135)	(1.152)	(1.151)	(1.120)	(1.117)	(0.959)	0.950
0.0	-5.453	-5.461	-2.944	-4.139	-2.764	-2.743	-0.590	-1.257	0.334	0.337
	(1.173)	(1.171)	(1.241)	(1.226)	(1.215)	(1.214)	(0.782)	(1.118)	(0.885)	0.882
0.4	-7.507	-7.513	-5.462	-5.463	-3.887	-3.869	-2.533	-2.527	-1.336	-1.33
	(1.302)	(1.300)	(1.325)	(1.322)	(1.253)	(1.252)	(1.108)	(1.105)	(0.856)	0.853
0.8	-10.047	-10.050	-6.503	-6.501	-4.690	-4.676	-3.449	-3.441	-2.491	-2.480
	(1.484)	(1.480)	(1.389)	(1.386)	(1.258)	(1.258)	(1.090)	(1.087)	(0.847)	0.844

Valores: $\hat{t}^*$

	Gauss	Urn22	Gauss	Urn22	Gauss	Urn22	Gauss	Urn22	Gauss	Urn22
θ|□	-0.8		-0.4		0.0		0.4		0.8	
-0.8	1.776	1.764	2.326	2.317	2.988	3.010	3.867	3.867	4.376	4.367
	(1.036)	(1.032)	(1.053)	(1.052)	(1.071)	(1.076)	(1.118)	(1.116)	(1.179)	1.166
-0.4	0.491	0.466	0.658	0.636	0.788	0.775	0.802	0.814	0.505	0.499
	(1.047)	(1.053)	(1.081)	(1.084)	(1.111)	(1.107)	(1.121)	(1.120)	(1.090)	1.083
0.0	-.0976	-.0984	-1.201	-1.155	-1.403	-1.383	-0.870	-1.727	-2.369	-2.366
	(1.086)	(1.085)	(1.227)	(1.229)	(1.133)	(1.134)	(0.869)	(1.113)	(1.051)	1.050
0.4	-2.386	-2.391	-2.917	-2.915	-3.301	-3.285	-3.744	-3.739	-4.569	-4.563
	(1.128)	(1.129)	(1.162)	(1.162)	(1.153)	(1.153)	(1.121)	(1.119)	(1.043)	1.043
0.8	-3.938	-3.936	-4.712	-4.706	-4.948	-4.934	-5.340	-5.333	-6.224	-6.216
	(1.209)	(1.213)	(1.198)	(1.198)	(1.161)	(1.162)	(1.119)	(1.118)	(1.045)	1.045

5.2 SIMULACION DE SISTEMAS

La aplicación al estudio de grafos Pert, mediante simulación Montecarlo, se incluye con un mayor detalle, en relación a las otras dos aplicaciones con sistemas complejos, con el propósito de comparar el funcionamiento de dos generadores diferentes.

En el modelo se establecen estimaciones, por simulación, de las medidas de la media y la varianza, relativas a la distribución de la "duración total de ejecución de un proyecto".

Se arriba a la conclusión que el generador lineal congruencial denominado UNR37, con respecto al generador basado en polinomios de Weyl, produce resultados que reflejan una mejor calidad aleatoria en las sucesiones de números aleatorios que se han utilizado alternativamente en el modelo.

En cuanto a las aplicaciones para resolver problemáticas de la región nord-patagónica argentina, como sistemas agrometeorológicos y forestales, la utilización de modelos de simulación se ha orientado principalmente al desarrollo de técnicas computacionales empleadas en los modelos que abarcan lenguajes de simulación para eventos discretos, y otros convencionales, numéricos y simbólicos.

Entre aquellas aplicaciones, se han incluido modelos de simulación de Precipitaciones de Granizo en la zona del Alto Valle de Río Negro y Neuquén; y modelos de evaluación de la eficiencia de distintas aeronaves en el Combate de Incendios Forestales, originados en la región Lacustre Precodillerana perteneciente a los Andes Patagónicos Argentinos.

5.2.1 UN ESTUDIO DE GRAFOS PERT MEDIANTE SIMULACION MONTECARLO

5.2.1.1 Introducción

En los problemas de ordenamiento que emplean redes de actividades, uno de los aspectos a resolver es la afectación de una duración a cada tarea mediante valores determinísticos (método CPM), o bien valores dependientes de una distribución de probabilidades.

La programación y control de proyectos de investigación y desarrollo utilizan frecuentemente el método PERT y la distribución Beta, por sus propiedades, a fin de asignar duraciones aleatorias a las tareas que conforman el proyecto representado mediante un grafo.

Este trabajo aporta una metodología para responder a las cuestiones: ¿entre qué límites varía el tiempo total mínimo requerido para la ejecución de un proyecto representado por un grafo Pert? ; y ¿cuál es la criticidad de las actividades representadas por los arcos del grafo?

Se ha adoptado un enfoque de simulación por Montecarlo para las estimaciones de la longitud total del proyecto, su varianza y la criticidad de las tareas, utilizando paquetes de programas actuales para el cálculo numérico y simbólico de expresiones tales como la función de distribución beta reducida.

Para el cotejo de los resultados con otras metodologías se adoptó en particular el mismo modelo de grafo Pert utilizado por G. Thomas (cfr.74).

Los resultados muestran ventajas en comparación con métodos similares y mejoras significativas con respecto al método clásico.

5.2.1.2 Generación de variables aleatorias de la distribución beta reducida

Es sabido que el Pert clásico utiliza una ponderación de tres tiempos para cada actividad (i, j): el tiempo optimista (a), mínimo requerido para que la tarea sea completada; el pesimista (b), tiempo máximo posible; y el más probable, que es el llamado tiempo normal requerido.

En las primeras investigaciones realizadas en Estados Unidos, buscando una distribución apropiada de la duración t_{ij} de una tarea, se fijaban ciertas propiedades implícitas en las definiciones precedentes, a las que se agregaba la referida al rango

o recorrido dado por la diferencia entre los números a y b , que debía ser seis veces el valor de la desviación típica, es decir: $(b-a)=6\square$

Sin embargo, (cfr.42) " ... al aceptarse con el Pert una ley Beta, se está dando una particular forma a la distribución a pesar de que no existen muchos estudios sobre la misma."

Si para algunas tareas del proyecto no existen estimaciones basadas en una suficiente experiencia, las estimaciones de a, b y m que se puedan proporcionar, resultarían incompatibles con las propiedades de la ley Beta.

Recordamos aquí que la ley de distribución de probabilidad Beta:

$$f(t)=k\,(t-a)^{p}(b-t)^{q}$$

así definida para valores de la variable independiente en el intervalo cerrado [a,b], y cero fuera del mismo, necesita la definición de los parámetros p y q.

$$f(t)=\left\{\begin{array}{c}\frac{\Gamma(p+q)}{\Gamma(p)\,\Gamma(q)}\;t^{\,p}\,(1-t)^{q}\quad,t\in[0,1];\;\;p,q>0\\ 0\quad si\;\;t\notin[0,1]\end{array}\right\}\;[1]$$

La función de distribución acumulativa para la variable aleatoria beta reducida o incompleta está dada por:

$$F(y)=I_{y}(p,q)=\frac{\Gamma(p+q)}{\Gamma(p)\,\Gamma(q)}\int_{0}^{y}x^{\,p}\,(1-x)^{q-1}dx\quad[2]$$

Algunos valores de I_y suelen encontrarse en tablas o en algunos paquetes estadísticos, particularmente para valores enteros de p y q.

En el modelo de nuestro trabajo los valores elegidos de los parámetros son:

$p=2+\sqrt{2}\;\;;\;q=2-\sqrt{2}$

Para la generación de variables aleatorias beta hemos elegido el método de la función inversa. Llamando U al número pseudoaleatorio, distribuido uniformemente en [0,1], se tiene:

$$\text{Si}\;\;F(X)=U\quad\text{entonces}\quad X=F^{-1}(U)\qquad[3]$$

siempre que F sea continua y monótona creciente.

Teniendo en cuenta la definición [2], X es solución de:

$$F(X)=\int_{-\infty}^{X}f\,(t)\,dt=U$$

o bien, empleando otra notación, X es solución de:

$$I_X\,(\,p+1, q+1) = I_X\,(3+\sqrt{2},\ 3-\sqrt{2}\,) = U$$

El procedimiento sugerido por Karian y Dudewicz (cfr.29) para obtener numéricamente la solución X en [3] es el método de Newton-Raphson:

$$x_{n+1} \;=\; x_n - \frac{g(x_n)}{g'(x_n)}\ \text{, para}\ \ g(x_n) = F(X_n) - U_i \quad [4]$$

El enfoque particular, que la resolución de [4] plantea, consistió en clasificar los valores de U obtenidos mediante un generador denominado URN37 (Dudewicz, Karian y Marshall, 1985), en dos subconjuntos (0;0.5] y (0.5; 1) para los cuales el valor inicial del método es 0.5 o 0.75, respectivamente.

Los buenos resultados de las pruebas aplicados al URN37 (cfr.56) y su adaptabilidad a los microcomputadores del tipo PC con 32 bits, justifican la elección de este generador.

Las variables beta generadas en la etapa subsiguiente verifican la prueba χ^2 para una muestra de 6'000 números y para un nivel de significación de $\alpha = 0.05$, lo que nos lleva a aceptar la hipótesis planteada, según la cual los datos provienen de una distribución beta.

De la ecuación [4] se obtiene X, una aproximación de la variable aleatoria beta, para un U dado. Efectuando entonces la siguiente transformación de variables:

$$X = \frac{V - A_{ij}}{B_{ij} - A_{ij}}\ \text{se obtiene}\ \ V = (B_{ij} - A_{ij})X + A_{ij} \quad [5]$$

donde A_{ij} ,B_{ij} , M_{ij} son las duraciones optimista, pesimista y más probable de la actividad (i,j), respectivamente.

En consecuencia, utilizando los valores X obtenidos por simulación para la ley beta reducida, procedemos a reemplazar la duración t_{ij} de cada tarea (i,j) por la variable aleatoria $V_{ij} = V$ de la expresión [5] anterior, en cada intervalo $[A_{ij}, B_{ij}]$.

El proceso de simulación fue ejecutado en un micro computador tipo PC. Para la programación se utilizaron el compilador Fortran MF386 NDP y el paquete matemático Maple V-Release 3, trabajando en forma interactiva para las corridas con los sistemas operativos Windows y DOS.

5.2.1.3 Un modelo de grafo Pert

Se considera una red de 30 nodos y 50 tareas de las cuales cuatro son ficticias (Figura N° 5.2.1). Para cada tarea (i, j) se calculan: $\square_{ij}$ y $\square_{ij}^2$, y luego, siguiendo el procedimiento clásico del método Pert, se replantea el modelo pero ya en forma determinística, asignando a la actividad (i, j) los valores antes calculados.

Luego se calcula el o los caminos críticos C, tomándose como media de la duración total T del proyecto a $\square_T$ (suma de los $\square_{ij}$); y como varianza $\square_T^2$ (suma de los $\square_{ij}^2$), para todo arco (i, j) de C.

Se admite, entonces, que la duración total T tiene una distribución normal N($\square_T$, $\square_T^2$).

5.2.1.4 Simulación de la longitud total del proyecto

Dado el grafo Pert descrito en el parágrafo 5.2.1.3, procedemos a aplicar el método para el cálculo del camino crítico, empleando una cantidad de corridas equivalentes a la longitud de la simulación, para lo cual se calculan dos estimaciones, de los parámetros media y varianza de T.

- Para la media E(T) tomamos como estimador: $\square = \frac{1}{N} \sum_{i=1}^{N} T_i$,

siendo N el número de simulaciones (a determinar).

Cuando N es grande, por aplicación del teorema central del límite a las variables muestrales T_i, se tiene: $\square_N = \frac{\varepsilon}{\frac{\square_T}{N}}$; $\varepsilon = \square - E(T)$

Para $\varepsilon < 0.5$ calculamos N para que la estimación de la media esté a menos del verdadero valor, ello con una probabilidad de 0.95.

Partiendo de los resultados que nos provee un soft convencional para el cálculo de caminos críticos se obtuvo: $\square_T = 186$, entonces, $\square_T^2 = 62$ o sea $\square_T \cong 8$.

y un camino crítico formado por los arcos (1,5); (5,14); (14,18); (18,28) y (28,30).

Tomando $\square_T = 8$ entonces $\frac{1}{15}\square_T \cong 0.5$ resulta:

$$P\left\{E(T) - \frac{\square_T}{15} < \square < E(T) + \frac{\square_T}{15}\right\} = 0.95$$

de donde:

$$P\left\{-\frac{\sqrt{N}}{15} < \square < \frac{\sqrt{N}}{15}\right\} = 0.95$$

Finalmente de la tabla normal se obtiene: $\Theta\left(\frac{\sqrt{N}}{15}\right) = 0.975$

lo que justifica la elección de N = 1'000

- Para la varianza se toma como estimador de $\square_T^2\,(T)$:

$$s^2 = \frac{1}{N}\sum_{i=1}^{N}(T_i - \square)^2$$

Según un teorema demostrado por R.M. Van-Slyke citado por Thomas, cuando N es bastante grande, el valor:

$$\square_N = \frac{\frac{N\,s^2}{\square^2(T)} - N + 1}{\sqrt{2N-2}}$$

sigue aproximadamente una ley normal reducida.

Bajo los mismos supuestos que Thomas adoptó, tomamos un ε' de modo que la estimación de la varianza esté a menos de ε' % del verdadero valor de $\square_T^2\,(T)$ con una probabilidad igual a 0.95.

Luego deducimos el valor de ε' cuando se selecciona N =1000, y luego, por tabla se obtiene: $\Theta\left(\frac{\frac{\varepsilon'_N}{100}+1}{\sqrt{2N-2}}\right) = 1.96$

Es decir que para N = 1'000 es ε' aproximadamente igual a 8.5

5.2.1.5 Criticidad de las tareas

Para cada actividad (i,j) del grafo se calcula su índice de criticidad c_{ij} definido como la razón entre el número de corridas para las cuales la tarea (i,j) es crítica, y el número total de corridas de simulación efectuadas.

Analizadas las frecuencias en mil corridas de la simulación se pudo constatar:

- Los índices de criticidad de las tareas (1,5) y (5,14) alcanzan al 70.7% ; la (28,30), el 57.7% ; las (14,18) y (18,28), el 44.9% .

- Que el camino (1,5) ; (5,14) ; (14,18) ; (8,28) ; (28,30) posee el índice más elevado de criticidad igual al 44.9 % , y es coincidente con el camino crítico del método tradicional.

- La existencia de caminos semi-críticos de duraciones próximas al crítico como los caminos:
(1,5) ; (5,14) ; (14,23) ; (23, 27) ; (27,30) con una criticidad del 11.5 % ;
(1,5) ; (5,14) ; (14,24) ; (24, 29) ; (29,30) con una criticidad del 9.5 % ;
(1,3) ; (3,9) ; (9,23) ; (23, 27) ; (27,30) con una criticidad del 8.6 % y
(1,3) ; (3,9) ; (9,24) ; (24, 29) ; (29,30) con una criticidad del 8.6 % .

En los restantes caminos no se observó valores de criticidad mayores que el cuatro por ciento.

5.2.1.6 Resultados

El Cuadro N° 5.2.1 muestra los resultados comparativos de la media y la varianza entre el método Pert, el método de simulación de Thomas, y el propio de simulación, donde se han determinado los respectivos intervalos con un nivel de confianza del 95%, para mil corridas.

Cuadro N° 5.2.1

Métodos	Método PERT	Simulación Thomas	Simulación Propia
MEDIA	200	[203 ; 204.8]	[205.67 ; 206.33]
VARIANZA	62	[16 ; 36]	[13.5 ; 30.4]

Del mismo surge claramente que en el tercer método, la varianza disminuye significativamente con respecto al método clásico, y está próxima al otro método de simulación, pero la media de la duración total es superior aplicando el último método.

Nuestros resultados referidos a la media y a la varianza son más precisos que el otro método de simulación. Para la media en particular, la ganancia en precisión puede atribuirse a que nuestro método de cálculo no incorpora el coeficiente de error $\varepsilon \leq 0.5$, que el autor mencionado adiciona a □ para estimar la varianza de la duración total T, de ejecución del proyecto, debido al uso de una función polinómica de aproximación para representar a la función de distribución; y por otra parte, a la mayor eficiencia del método congruencial URN37 (Dudewicz, Karian y Marshall), con respecto al generador basado en los polinomios de Weyl empleado por Thomas.

Además, este resultado es coincidente con las conclusiones del estudio ya realizado al abordar la construcción de sucesiones originadas en tales polinomios, que emplean cifras de números irracionales (cfr.3.2.3 y Cuadros Nº 4.2.4.5 y 4.2.4.6).

De este modo, para el grafo Pert analizado, queda respondido el interrogante inicial sobre los límites de variación de la duración total de ejecución de un proyecto bajo las hipótesis adoptadas.

Y la respuesta al segundo interrogante queda expresada una vez determinados los índices de criticidad de las tareas del grafo que aportan al tomador de decisiones un instrumento en el seguimiento de las etapas críticas para su verificación, control y eventual reprogramación del proyecto.

Figura N° 5.2.1 Grafo Pert

5.2.2 SIMULACION DE MODELOS POR PRECIPITACIONES DE GRANIZO

5.2.2.1 Introducción

La importancia del fenómeno granizo para la economía regional del Alto Valle de Río Negro y Neuquén, y la necesidad de adoptar métodos de prevención, dieron lugar a la participación de la Universidad Nacional del Comahue, en un proyecto del cual el presente constituye una primera etapa en los estudios acerca del origen, trayectorias y modalidades de las precipitaciones de granizo.

A partir de datos recogidos y elaborados entre 1966 y 1986 (cfr. 35), se formularon distintos modelos de simulación teniendo en cuenta variables tales como el potencial frutícola; la cantidad y tipo de tormentas de granizo; el estado fenológico de las plantaciones; y algunas variables económicas como los precios de mercado.

En el sistema granizo las componentes analizadas fueron: las tormentas y las localidades con sus respectivos atributos, los tipos de tormentas, según la superficie que afectaran; las características meteorológicas; y el potencial frutícola representado por manzanas y peras.

Las variables aleatorias intervinientes en el modelo elaborado representan: el número de tormentas ocurridas en un mes dado (0 a 4); el tipo de tormenta (I a IV), según un esquema conceptual basado en analogías con el utilizado por el modelo europeo (cfr.1); la localidad afectada (en total veintiuna con jurisdicciones en ambas provincias); y la superficie correspondiente (tres tipos de daños según superficie y densidad del granizo).

La utilización de un lenguaje orientado a simulación (GPSS-PC), y otros convencionales con interfaces de rutinas incorporando generadores propios, permitió alcanzar resultados que, según las pruebas estadísticas, son aceptables con un grado satisfactorio de confiabilidad, confirmando tendencias históricas.

5.2.2.2 Modelos

5.2.2.2.1 Sistema Granizo

A partir de los análisis de los fenómenos meteorológicos, llevados a cabo por el grupo del proyecto central de lucha antigranizo, las primeras etapas se orientaron a diseñar distintos modelos que explicasen el comportamiento espacio-temporal del "Sistema Granizo", el que se esquematiza en la Figura N° 5.2.2.1

La investigación espacio-temporal de las precipitaciones registra los más recientes avances en las campañas realizadas en Grossversuch (Suiza) a través de un proyecto entre Francia, Suiza e Italia (década del setenta), con el objetivo de verificar la eficacia del método soviético de prevención de las granizadas, mediante inseminación de tormentas por cohetes.

Con la utilización de una buena información provista por radares y redes de impactómetros se definieron la "forma", la "naturaleza" y la "dinámica" de las células de tormentas.

Del estudio comparado sobre numerosas observaciones en Napf (Suiza), Alberta (Canadá) y Transval Highveld (Sudáfrica), surge un esquema conceptual unitario explicativo, que muestra notables analogías con la región del Alto Valle de Río Negro y Neuquén.

Los tres conceptos esenciales en que se basa el modelo europeo son:

- Una precipitación de granizo no es uniforme en el espacio.
- Los tipos de precipitaciones (1, 2 y 3) sólo constituyen tres aspectos de una forma única que evoluciona de un estado primitivo (tipo 1) hasta un estado terminal (tipo 3).
- La evolución es continua del estado uno al tres; y el estado cuatro, para tormentas muy grandes, es simplemente una repetición del estado tres.

Para la determinación de las formas y estructuras de las superficies de granizo se toman en consideración: la repartición espacial de las densidades, la cantidad, diámetro, masa y energía cinética.

Se concluye en la determinación de formas rectangulares alargadas en el sentido de las trayectorias de las tormentas.(Modelo europeo, Figura N° 5.2.2.2).

La repartición preferencial del granizo aparece en las zonas centrales, mientras que los valores más bajos están en la periferia, disponiéndose en dos zonas concéntricas (tipo 2). Para tormentas del tipo tres, el fenómeno de concentración se

acentúa bajo la forma de tres zonas rectangulares concéntricas. El caso de tormentas muy grandes (tipo 4) no presenta caracteres nuevos y se aproxima mediante repeticiones de la forma del tipo tres.

En el Cuadro Nº 5.2.2.1, a los tipos de superficies señalados, se agregan las superficies medias, el porcentaje de daño según superficie y cuantificados por analogía con el modelo europeo y las componentes S_{ij} de daños.

Una vez determinados los valores de los distintos daños D_{i0}, se cuantificaron en unidades monetarias (u$s) utilizando el modelo económico de precios históricos de manzanas y peras, desarrollado en la publicación (cfr.35).

Para ilustrar una precipitación que explique el modelo de daños, consideraremos como ejemplo una tormenta tipo II. El daño que provoca, puede describirse como formado por dos partes: una en el centro (S22) y otra en la periferia (S21). En la parte central, el daño es mayor alcanzando el 10%, mientras que en la zona S21 el daño solamente alcanza el 5%.

A continuación se indican las expresiones matemáticas que definen los daños según los diferente tipos:

$Daño10 = 0.05 * S10$
$Daño20 = Daño21 + Daño22 = 0.05 * S21 + 0.3 * S22$
$Daño30 = Daño31 + Daño32 + Daño33 = 0.05 * S31 + 0.3 * S32 + 0.7 * S33$

En el programa, se elige al azar una localidad afectada, y luego se efectúa la búsqueda de las localidades vecinas. Para ilustrar el método empleado se considera el siguiente ejemplo: si la localidad dañada fuese Cipolletti, y la superficie que abarca la tormenta supera el área real de ese municipio, entonces se toma la diferencia entre la longitud de Cipolletti y la longitud de la tormenta. Luego se distribuye el área restante entre las localidades vecinas, continuando con la primera que se encuentre al Este de Cipolletti (F. Oro), y así siguiendo.

Es de hacer notar que podrían surgir restricciones en el modelo, cuando no se toman en consideración los efectos de precipitaciones importantes ocurridas en zonas precedentes a las cinco primeras localidades, ya que no se posee información sobre la zona de bardas aledañas al Alto Valle.

Algunas hipótesis, que fueron consideradas como básicas, se enuncian seguidamente; unas se confirmaron, y otras están sujetas a estudios específicos que se continúan para su verificación:

1. Determinación de la temporada de granizo Octubre - Marzo.

2. Las variables: "cantidad de tormentas por mes" y por "localidad" son estadísticamente independientes.
3. Las ocurrencias de tormentas son variables aleatorias con propiedades de procesos markovianos.
4. Una misma tormenta puede subdividirse por localidades.
5. El potencial frutícola es uniforme por localidad.
6. Delimitación a la región del Alto Valle.
7. Dirección de las precipitaciones: Oeste - Este
8. Alcance aleatorio del número de localidades afectadas, según el tipo de tormenta.
9. Formas rectangulares de las superficies dañadas.
10. Independencia con relación a los ciclos solares.

5.2.2.2.2 Modelos computacionales

Básicamente el proceso de simulación sigue la ilustración de la Figura N°5.2.2.3

Para generar aleatoriamente el número de tormentas, de (0 a 4), y el tipo (para el caso que el número de tormentas esté entre 1 y 4), se consideraron las frecuencias aportadas por datos históricos.

En los primeros programas confeccionados en un lenguaje de propósitos generales donde se incluyó también el sorteo de un número de localidades vecinas a la afectada; posteriormente esta característica fue determinada con más precisión incorporando datos sobre medidas de longitud de las localidades y coeficientes económicos, por unidades de superficie cultivadas en cada una de ellas. De ese modo, según el tipo de tormenta sorteado, se calcularon las áreas afectadas, también de modo aleatorio, y consecuentemente la evaluación de daños en manzanas y peras.

Conforme al modelo diseñado el cálculo de superficies sigue las ecuaciones de 5.2.2.2.1 y la búsqueda de las localidades adyacentes a la sorteada toma la dirección aproximada oeste-este.

5.2.2.3 Variables de los modelos

Como es sabido la variable "tipo de tormenta" se relaciona con la extensión de la tormenta, el tamaño y densidad del granizo.

El objetivo principal es estimar el número medio de tormentas por temporada, para cada tipo, confeccionando intervalos de confianza para dichos parámetros, los que fueron confrontados con los resultados de la simulación, con el fin de validar esta información artificial.

Se generaron veinte temporadas de granizo y se contaron las cantidades de tormentas de cada tipo; con esa información se calculó el número medio de tormentas y el desvío estándar, confeccionando intervalos de confianza para los parámetros poblacionales.

Se conoce por un estudio previo de este proyecto (cfr.58) que la cantidad de tormentas por tipo, en el Alto Valle, se ajusta muy satisfactoriamente a una distribución Poisson, con parámetros diferentes para cada mes de la temporada Octubre - Marzo (Figura N°5.2.2.4).

Analizando la distribución geográfica de la variable "cantidad de tormentas por localidad", y ordenando las localidades a lo largo del eje del Alto Valle, se constata que no es uniforme debido a la existencia de diferencias que no han sido aún explicadas por la ausencia de estudios meteorológicos. (v.gr. la región no cuenta entre otros recursos con radar que permita determinar características de la física de las nubes).

Sin embargo, se observa que existen dos zonas con mayor frecuencia: una en la confluencia de los ríos Limay y Neuquén, que afecta aproximadamente a las localidades de Cipolletti y Allen; otra, al terminar el Valle, donde se encuentran Villa Regina y Chichinales, lo que se muestra en la Figura N° 5.2.2.5: "Localidades región Alto Valle de R. Negro y Neuquén".

En cuanto a las pérdidas, se han estimado siguiendo el modelo de precios históricos de la fruta. Su elaboración por miembros del proyecto central contempla las fluctuaciones del precio de la fruta, a nivel productor, obteniéndose un valor promedio (U$S 0.11 el kg. de peras y U$S 0.14 el kg. de manzanas). Otro factor que se tuvo en cuenta es la distribución de las densidades de granizo en la tormenta.

De las corridas efectuadas se obtiene un promedio de pérdidas para:

Manzana: $ 6.872.010 ; Pera: $ 1.466.176 ; Total: $ 8.338.186

5.2.2.4 Conclusiones

1. Los daños económicos fluctúan entre temporadas sin ocurrencia de tormentas, por consiguiente, sin pérdida; y otras, con pérdidas cuantiosas. Se determina en general que el daño, en millones de dólares, causado a las manzanas fue de $ 6.872.010, mientras que a las peras fue de $ 1.466.176.

En consecuencia se calcula en promedio una pérdida total de $ 8.338.186, entre ambos tipos de frutas en el Valle de Río Negro y Neuquén.

2. La cantidad de tormentas de granizo en la temporada Octubre - Marzo sigue aproximadamente la ley de distribución de probabilidades poissoniana.

3. Entre los tipos de tormentas se distinguen dos subgrupos, uno formado por los tipos I y II, y el otro por los tipos III y IV. Los primeros representan alrededor del 85% de las precipitaciones, siendo el tipo I el más frecuente entre ellos (49%). Del segundo grupo el de mayor ocurrencia es el de tipo III (11%).

4. Las localidades más afectadas son: General Roca, Villa Regina, Allen y Cipolletti.

Teniendo en cuenta los datos de la muestra histórica de veinte temporadas que entre 1966 y 1986 afectaron la zona del Alto Valle de Río Negro y Neuquén, a cada experimento de simulación se le ha asignado la duración de una temporada, puesto que uno de los objetivos es utilizar el modelo para realizar pronósticos anuales.

Es necesario destacar que los resultados obtenidos en esta investigación están justificados en datos históricos, pero no actualizados a la fecha.

Por otra parte, en algunos casos, se empieza a utilizar cañones antigranizo que dispersan y desvían las tormentas; en otros, los sistemas de redes, disminuyendo así el efecto que el fenómeno granizo causa a los cultivos de la región.

Cuadro N° 5.2.2.1 Modelo de tormentas

Tipo	**Sup. med.**	**S_m<S_{io}<S_n**	**Daño**	**S_{io}**
I	8.60	< 25 km^2	3%	$S_{10}=L^2/3$
II	35.72	25 - 50 km^2	11%	$S_{20}= S_{21}+ S_{22}$
III	55.86	50 - 100 km^2	17%	$S_{30}= \sum S_{ij}$
IV	348.33	> 100 km^2	35%	$N*S_{30}$

Figura N° 5.2.2.1 Sistema granizo

Figura N° 5.2.2.2 Modelo europeo

Figura N° 5.2.2.3 Proceso de simulación

Figura N° 5.2.2.4 Distribución de tormentas por tipo

Figura N° 5.2.2.5 Localidades región Alto Valle de R. Negro y Neuquén

5.2.3 EVALUACION DE RENDIMIENTOS DE DISTINTAS AERONAVES PARA COMBATIR INCENDIOS FORESTALES

5.2.3.1 Introducción

Uno de los primeros aspectos que en el presente trabajo se ha abordado fue el estudio de las principales causas que afectan la eficiencia de una aeronave en la lucha contra incendios forestales.

Con tal objetivo se han incluido en los respectivos modelos las características técnicas de las aeronaves, los atributos geográficos de la zona en la cual operan, la cantidad de litros de agua que deben arrojar, y también la cantidad de incendios simultáneos que pueden producirse. Estos estudios son presentados en la primera parte del trabajo.

Fundamentalmente, se intenta evaluar, en una primera etapa la fiabilidad de una flota de pequeñas aeronaves frente a una sola aeronave bajo iguales capacidades hidrantes y los mismos valores en costos de adquisición y operativos.

Para facilitar un análisis comparativo entre la flotilla de pequeños aviones y una aeronave de dimensiones mayores, se han desarrollado varios modelos siguiendo un orden creciente de complejidad. Posteriormente se incorporan otras aeronaves -segunda etapa del trabajo- como son los helicópteros y los fumigadores.

Los resultados ponen de relieve la baja fiabilidad de contar con un sistema de apoyo de una sola aeronave en la lucha contra incendios forestales; y por otra parte, en todos los casos en que los incendios sean medianos o grandes, el mejor rendimiento de una flota de hidroaviones pequeños, frente a las otras alternativas.

5.2.3.1 Estudio de algunas variables

5.2.3.1.1 Las aeronaves

Se consideran inicialmente dos aeronaves: un hidroavión de origen canadiense, el denominado CL-215 Canadair, y un pequeño hidroavión bombero, cuyo anteproyecto de diseño ha sido elaborado en nuestra Universidad con una configuración biplana.

El avión CL-215, tiene una capacidad hidrante de 6000 litros y una autonomía de vuelo igual a la del pequeño hidroavión (240 minutos). Características tales como la velocidad de crucero, velocidad de ascenso y otras que describen el vuelo, son levemente superiores en el CL-215.

Otras características son: capacidad hidrante de 1000 litros; tiempo de ascenso (para 600 metros) igual a 180 segundos; distancia horizontal en el ascenso, 40 m/seg de 7200 m; velocidad crucero de 250 km/h igual a 69.44 m/seg; tiempo de carga del agua, 60 segundos; tiempo nulo de descarga del agua; carga agua desde el lago; tiempos de ascenso desde la base y desde el lago son iguales a los del descenso; las distancias horizontales en el ascenso desde la base y desde el lago son iguales a las de descenso; y con las características de ser anfibio y poder aterrizar en caminos de tierra.

Con posterioridad, en el proyecto de investigación se incluyeron otros tipos de aeronaves: los helicópteros y los fumigadores.

El helicóptero posee algunas características como las siguientes: capacidad hidrante de 500 litros; tiempo de ascenso de 75 segundos (para 600 m); distancia horizontal en el ascenso de 0 metro; tiempo de descenso y distancia horizontal de descenso idénticos a la del pequeño avión bombero; una velocidad crucero de 210 km/h; tiempo de carga del agua, de 30 segundos; tiempo nulo de descarga de agua; carga agua desde el lago.

El avión fumigador tiene características similares al anterior, difiriendo en cuanto a capacidad hidrante, de 600 litros; distancia horizontal en el ascenso (yendo a 40 m/seg) de 7200 metros; tiempo de carga del agua, igual a 600 segundos; la carga del agua se realiza en la base. No hay restricciones en cuanto a las distancias que deben existir desde las bases al foco de incendio (a diferencia de la restricción de 150 km, considerada en los otros dos modelos).

5.2.3.1.2 Las características geográficas

El ámbito geográfico comprende la Región de los Andes Patagónicos Argentinos, particularmente la zona cordillerana que se extiende desde el paralelo 39 hasta

el 44; dicha región es una larga franja de bosques y presenta como característica sobresaliente gran cantidad de lagos y lagunas aptos para ser utilizados por los aviones bomberos para la carga de sus tanques de agua.

La región puede ser esquematizada como un rectángulo de 550 km de largo por 50 km de ancho que fue dividido en cinco sectores iguales. A través de una carta geográfica fueron determinadas las áreas conformadas por los lagos y luego mediante digitalización se discretizaron sus formas.

En lo que respecta a aeródromos y aeropuertos aptos para la operación de aviones, se seleccionó al Aeropuerto de San Carlos de Bariloche como base de operaciones para los modelos en que interviene el CL-215; mientras que para aquellos modelos donde participa la flotilla de aviones pequeños, se eligieron como aptos seis aeródromos: Aluminé, San Martín de los Andes, San Carlos de Bariloche, El Bolsón, Esquel y Corcovado.

5.2.3.1.3 Capacidades hidrantes

El estudio de la cantidad de litros de agua arrojados proporciona un conocimiento acerca del trabajo que deben efectuar las aeronaves para realizar tareas de combate de incendios, de refresco, etc. La función empírica de distribución de agua, se ha obtenido utilizando información recopilada por la Fuerza Aérea Española en 42 incendios donde intervinieron los aviones canadienses.

Se deduce de dicha curva que aproximadamente en el 74 % de las acciones hubo que arrojar a lo sumo 54000 litros, es decir solo se registra un 26 % de incendios para los cuales es necesario arrojar mayores cantidades de agua.

5.2.3.1.4 Los incendios simultáneos

Denominaremos así a dos o más incendios que se producen el mismo día. De acuerdo a datos obtenidos de los registros de incendios en las provincias de Río Negro y Neuquén, encontramos que un 30 % de los incendios son simultáneos. Esta proporción, como se verá mas adelante, incide fuertemente en la eficiencia de los aviones, principalmente sobre el CL-215.

5.2.3.2 Descripción de los modelos

5.2.3.2.1 Avión canadiense sin incendios simultáneos (modelo I)

Enunciaremos las hipótesis de este modelo:

1. Se considera un único incendio diario.

2. Se supone que el avión tiene como base el Aeropuerto de San Carlos de Bariloche.

3. Se emplean la velocidad de crucero, de ascenso, deslizamiento y descarga seca los valores proporcionados por el fabricante de la aeronave.

4. Se caracteriza la eficiencia de acuerdo a la clasificación de Carrasco-Gil (cfr. 8).

En este modelo se generan al azar las coordenadas del incendio y luego se calculan las distancias entre la base, el fuego y el lago más cercano a este último. Posteriormente, se genera en forma aleatoria la cantidad de litros de agua que deberá arrojar el avión y con estos datos se determinan los tiempos de vuelo para finalmente registrar estadísticas y hacer cálculos de eficiencia.

5.2.3.2.2 Modelo flotilla de pequeños aviones sin incendios simultáneos (modelo II)

Las hipótesis utilizadas en este caso son:

1. Se considera un único incendio diario.

2. Se seleccionan seis bases de operaciones, cada una de las cuales posee un avión. Los aeródromos utilizados son Aluminé, San Martín de los Andes, Bariloche, El Bolsón, Esquel y Corcovado.

3. Se emplean valores de velocidad de crucero, velocidad de ascenso y otros proporcionados por el equipo de diseño del pequeño hidroavión bombero de la Universidad Nacional del Comahue.

4. Se consideran en condiciones de combatir un incendio todas las aeronaves que se encuentran a una distancia no mayor de 150 km del incendio.

5. Las aeronaves parten simultáneamente de las bases de operaciones.

En general la lógica es análoga a la utilizada para el modelo anterior, con algunas diferencias, como por ejemplo, las de considerar en este caso la cantidad y los tiempos de vuelo de los aviones que intervienen, para determinar los momentos en que cada uno de ellos arriba al incendio y el tiempo disponible con una, dos o más aeronaves para combatir el incendio.

El algoritmo contiene como etapas:

1. Generación de coordenadas de incendio según zona:

Cálculo de coordenadas del incendio; determinación de la zona donde se generó el incendio, elegida aleatoriamente en base a una función.

2. Calcular los parámetros:

Coordenadas de las bases; distancia y número de lagos más próximos al foco del incendio; bases distantes del lago más próximo, a lo sumo 150 km; distancia foco incendio a base más próxima; valores de retorno.

3. Cálculo de cantidad de aviones y tiempos de arribo al foco:

Verificar la cantidad de aeronaves disponibles para combatir el incendio; calcular los tiempos de arribo al foco de los distintos aviones.

4. Generación de tipos de incendio:

Generar aleatoriamente el tipo de incendio; identificar la cantidad de litros de agua por arrojar; determinar la cantidad de vuelos.

5. Clasificación de eficiencia:

Calcular los tiempos de operación y de vuelo para los distintos aviones; clasificar la eficiencia con los conceptos: Optimo-Bueno-Regular-Malo.

Luego de efectuar las pertinentes corridas de los modelos de simulación I y II, se observa que el avión canadiense posee una eficiencia de aproximadamente un 80 % (óptimo y bueno); esto se explica por la gran cantidad de lagos que presenta la geografía de la región. De este modo la aeronave tiene, en la mayoría de los casos, acceso a un lugar cercano al incendio donde se reaprovisiona de agua, y así se reduce sensiblemente el tiempo entre lanzamientos sucesivos de líquido sobre el fuego.

La Figura N°5.2.3.1 muestra los resultados obtenidos cuando se consideran incendios simultáneos, ya que la aeronave canadiense puede atender un único siniestro, es decir que deja sin apoyo aéreo a los otros incendios, y por ello su eficiencia decae notablemente, en comparación con el modelo de un incendio diario.

Por otra parte, la flotilla de pequeños aviones disminuye levemente su eficiencia con respecto al correspondiente modelo de un incendio, pero en contraste con el avión canadiense tiene mejor rendimiento. Incluso en los casos que resulta ineficiente, es preferible la flotilla pues no deja a ningún incendio sin apoyo aéreo.

5.2.3.2.3 Modelos helicóptero (III) y avión fumigador (IV)

Las siguientes hipótesis son comunes a los modelos I, III y IV:

1. En cada base hay una aeronave.

2. Al foco de incendio convergen las aeronaves cuyos lagos se encuentran a una distancia no mayor de 150 km del mismo, con excepción de los fumigadores, como ya fue mencionado.

3. Los helicópteros e hidroaviones cargan agua en los lagos más cercanos a los focos de incendio, mientras que los fumigadores cargan agua en las bases más

cercanas a los focos.

4. Las aeronaves parten de las bases de operaciones, al mismo tiempo.

Como ejemplo más detallado de uno de los modelos, modelo III (helicóptero), se describen las variables y sus respectivas ecuaciones:

D1: Distancia de carreteo y elevación.

D1 = 0, pues no tiene carreteo.

DT1: Tiempo de elevación.

DT1= 2.5 minutos

DT4: Tiempo de deslizamiento sobre el agua.

DT4: 0.5 minutos

D5: Distancia de ascenso desde el lago.

D5 = 0, ya que no tiene carreteo.

DT5: Tiempo de ascenso desde el lago.

DT5 = 300m /(2.5m /seg) = 2 minutos.

BMPF: Base más próxima al foco.

D2A: Distancia de la BMPF al incendio.

D2A = BBB

DT2A: Tiempo de vuelo crucero de la BMPF al incendio.

DT2A = (D2A * 60)/210

DT2B: Tiempo de vuelo crucero de la segunda BMPF al incendio.

D2B = DDD

DT2B=(D2B*60)/210

DT2C: Tiempo de vuelo crucero de la tercera BMPF al incendio.

D2C=EEE

DT2C=(D2C*60)/210

DT2D: Tiempo de vuelo crucero de la cuarta BMPF al incendio.

D2D=FFF

DT2D=(D2D*60)/210

DT2E: Tiempo de vuelo crucero de la quinta BMPF al incendio.

D2E=GGG

DT2E=(D2E*60)/210

D6 = Duración del vuelo crucero lago-foco

D6 = AAA - Ascenso lago-foco (D5)

DT6 = Duración del vuelo crucero lago-foco

DT6 = (D6*60)/210

AGU=LTU*500

IBASE =Tiempo crucero de la base al incendio

5.2.3.3 Conclusiones

1. Las características geográficas inciden fuertemente en la elección del tipo de aeronave a ser utilizada para apoyo aéreo en incendios forestales.

2. La eficiencia de una aeronave de gran porte es menor que la observada para la flotilla de pequeños hidroaviones. Esta diferencia se acentúa principalmente cuando se consideran incendios simultáneos (Figura Nº5.2.3.1).

3. La localización de las bases de operaciones para las aeronaves es una variable de considerable influencia en la optimización del sistema.

4. El hidroavión es el modelo que mejor desempeño posee en comparación con los otros modelos.

5. Los tiempos de mayor incidencia en la medida de eficiencia son los de circuito y de operación total. En el caso del fumigador son notablemente superiores, ya que la trayectoria de esta aeronave es de la base al foco (y no del lago al foco) cuyas distancias son superiores debido a la zona geográfica considerada.

6. De la comparación entre hidroavión y helicóptero, que poseen un rendimiento similar, el primero posee una mayor capacidad hidrante, lo que le permite disminuir el tiempo de circuito, y el helicóptero lo reduce por no necesitar carreteo.

Cuadro N° 5.2.3.1 Comparación de rendimientos

Tabla	Hidroavión	Helicóptero	Fumigador
Agua	24'000	12'000	14'400
Demor1	22	21	25
Demor2	27	51	33
Demor3	46	77	55
Demor4	95	110	112
Demor5	6	8	130
Circuito	7	8	28
Eficiente	1	2	4
TVuelo	253	325	514

Figura N° 5.2.3.1 Eficiencia avión grande vs. flotilla

Cuadros N° 5.2.3.2 y 5.2.3.3 Simulaciones de incendios

Las coordenadas del incendio son: latitud 33, longitud 196
El tamaño del incendio es: 491.923
El lago elegido es: D4
La base elegida es: Esquel
La cantidad de helicópteros es: 1
La cantidad de viajes de cada uno es: 50
El tiempo requerido es (min.): 156.52248
La cantidad de fumigadores es: 1
La cantidad de viajes de cada uno es: 29
El tiempo requerido es (min.): 492.31485
La cantidad de hidroaviones es: 1
La cantidad de viajes de cada uno es: 18
El tiempo requerido es (min.): 140.35031

La cantidad de incendios chicos fueron: 3046
La cantidad de incendios medianos fueron: 3487
La cantidad de incendios grandes fueron: 3467
Para incendios de 50 a 500 m2:
Helicópteros: 26.0%
Fumigador: 0.0%
Hidroavión: 74.0%
Para incendios de 500 a 1000 m2:
Helicópteros: 34.0%
Fumigador: 0.0%
Hidroavión: 66.0%
Para incendios de 1000 a 1500 m2:
Helicópteros: 0.0%
Fumigador: 0.0%
Hidroavión: 100.0%

IBASE = (CCC*60)/210

A fin de almacenar los resultados registrados en el proceso de simulación, se han empleado distintas tablas:

AGUA: Tabla que registra los litros de agua arrojados.

CIRCUIT: Tabla que registra tiempos de circuitos de cada avión.

DEMOR 1: Tabla que registra los tiempos de arribo del primer avión.

DEMOR 2: Tabla que registra los tiempos de arribo del segundo avión.

DEMOR 3: Tabla que registra los tiempos de arribo del tercer avión.

DEMOR 4: Tabla que registra los tiempos de arribo del cuarto avión.

DEMOR 5: Tabla que registra los tiempos de arribo del quinto avión.

EFICIEN: Tabla que registra la eficiencia de cada avión en el apagado del incendio.

TVUELO Tabla que registra el tiempo total de operación de los aviones.

En el Cuadro Nº 5.2.3.1 se muestran los valores medios correspondientes al rendimiento de tres tipos de aeronaves: hidroavión, helicóptero y fumigador.

Una reciente actualización de estos modelos, utilizando programas en lenguaje JAVA, permite incorporar funciones aleatorias, que además de generar las coordenadas de los incendios, calculan el tamaño del incendio en metros cuadrados; los tiempos empleados para la carga de combustible en las aeronaves; y proporcionan estadísticas de porcentajes efectividad, como así también la facilidad de ejecutar una corrida para analizar cuál fue la base elegida, lugar y magnitud del incendio, lago seleccionado para abastecimiento de agua, y los tiempos totales insumidos por cada aeronave.

Los resultados correspondientes a los casos de una y diez mil simulaciones se muestran en los Cuadros N° 5.2.3.2 y 5.2.3.3.

6. CONCLUSIONES Y RECOMENDACIONES

6.1 OBJETIVOS DEL CONOCIMIENTO

Los objetivos del conocimiento en nuestro trabajo de la tesis *Generación de números pseudoaleatorios eficientes en microcomputadores,* pueden resumirse en los siguientes puntos:

a) Estudio y obtención de un nuevo conjunto de generadores congruenciales (GNA) para producir sucesiones de números pseudoaleatorios con las siguientes propiedades, en orden de importancia: ser reproducibles y poseer períodos suficientemente largos, rapidez y poco consumo de memoria, sencillez de programación y portabilidad; y la incorporación de nuevas técnicas de programación que sean compatibles con las más usuales arquitecturas de PC.

La pregunta primaria que respecto a este punto nos planteamos es:

¿Existe en ámbitos académicos un paquete ejecutable en microcomputadoras que contenga un conjunto de generadores de números pseudoaleatorios elaborados con propósitos de docencia e investigación, en castellano, con amplia documentación y disponible sin cargo?

En el cap.4), parágrafo, **4.1 Los generadores,** se responde satisfactoriamente a la pregunta, donde el tema GNA constituye una parte del sistema creado a estos fines: GENTSTPC.

b) Análisis de las **bibliotecas de pruebas estadísticas teóricas y empíricas** existentes destinadas a la verificación de la calidad de las sucesiones de números pseudoaleatorios creados por los GNA, y **creación de nuevas pruebas**.

La pregunta inicial fue:

¿ Es posible disponer de bibliotecas de pruebas tradicionales que puedan instalarse y ejecutarse en equipos del tipo PC, con sistemas operativos muy accesibles y procesadores de 32 bits **para evaluar sucesiones de números** pseudoaletorios; y también, poder incorporar nuevas baterías de pruebas para los mismos propósitos y entorno operativo?

En el cap.4), parágrafos **4.1 Pruebas empíricas en Gentstpc,** y **4.3.1 Pruebas basadas en un estimador de entropía,** se brinda una solución al problema formulado en la pregunta anterior.

Las pruebas de entropía son un aporte nuevo y siguen algunas de las ideas del artículo, en curso de publicación: *"Evaluation of RNG using an estimator of entropy" by Angel Adolfo Olmos and María C. Allan, submitted to the American Journal of Mathematical and Management Sciences, 4-96)*

c) Accesibilidad a un paquete de programas compatible con microcomputadores(PC) y que emplean sistemas operativos ampliamente difundidos.

Los objetivos enunciados en a), GNA, y b), Pruebas de sucesiones de números pseudolaleatorios, se plasmaron en un solo sistema:GENTSTPC (GEN, generadores; TST, tests; PC, microcomputadores), detallado en Cap.4. Gentstpc fue desarrollado para dos sistemas operativos (D.O.S. y Windows) en 131 subrutinas y 6 subprogramas funciones, totalizando 18'000 líneas de códigos de programas, que se presentan en cuatro archivos ejecutables contenidos en un soporte magnético, disco flexible de 1.44 MBytes, de simple instalación y ejecución.

En el Anexo II, distribuido en un CD-ROM, se incluyen: Manual del Usuario, una extensa documentación de los programa-fuente, y variados ejemplos explicados.

d) Establecer una **nueva metodología para la selección de un GNA eficiente**

A partir de las cuestiones formuladas en los objetivos a) y b) se han integrado ambas soluciones para determinar condiciones de eficiencia de un GNA. La metodología propuesta consta básicamente de dos etapas de pruebas empleando Gentstpc: en la primera, se pasan 19 pruebas estadísticas empíricas conocidas, y a sus resultados se les aplica reiteradamente Chi-cuadrado y Kolmogorov-Smirnov, con un número de sucesiones y lotes que pueden alcanzar hasta diez millones de números pseudoaleatorios, por cada GNA probado. En la segunda etapa se aplican las pruebas de entropía. El sistema Gentstpc dispone también de cuatro pruebas teóricas que pueden confirmar la selección obtenida en la metodología.

Las tres alternativas para llevar a cabo la Metodología son: correr el ejecutable Gentstpc1 (en ambiente Windows); Gentstpc (bajo DOS); y el ejecutable Gentstpc2 (bajo Windows) para crear archivos de texto formados por hasta dos millones de números producidos por los GNA del paquete, denominados URN**, los que posteriormente pueden ser verificados mediante las pruebas ya mencionadas.

e) Obtención de números **pseudoaleatorios generados con un método no congruencial (Polinomios de Weyl)**

La cuestión inicial planteaba:
¿Es posible modelar un método de GNA con un algoritmo basado en polinomios de Weyl, mediante una arquitectura de 32 bits?

Partiendo de una aproximación racional a un número irracional (por ej. □, *e*) y seleccionando una base numérica, la cantidad de cifras de cada número a generar, y el número de las respectivas sucesiones independientes, se arribó a **formular un algoritmo (Poliweyl)** de sencilla programación y compatible con la arquitectura propuesta.

f) Pruebas de **hipótesis de calidad de un generador basado en polinomios de Weyl**

En el estudio de distintas sucesiones de números pseudoaleatorios generados por polinomios (Weyl), incorporando semillas irracionales a través de cálculos de miles de sus cifras (*e* y □), se ha comprobado que fallan en algunas propiedades de uniformidad.

g) Propiedad de contrastar las propiedades estadísticas de las sucesiones de **números pseudoaleatorios producidos por los generadores de otros paquetes o lenguajes, o por nuevos algoritmos.**

Gentstpc posee una opción (bajo Windows) que permite realizar estudios de calidad estadística de las sucesiones de números pseudoaleatorios producidos por otros paquetes, a través de archivos tipo texto, cuyos datos son obtenidos ya sea mediante rutinas específicas de los sistemas analizados, o bien como salidas en un formato adecuado, en el caso de nuevos generadores codificados en otros lenguajes y en otros sistemas operativos y plataformas.

Desafortunadamente, aún hoy es posible encontrar ciertos sistemas computacionales que incorporan, vía rutinas por defecto, unos GNA que no han sido cabalmente verificados, y que por lo tanto no son recomendables para determinadas aplicaciones. En el parágrafo 5.1 se amplían algunos resultados hallados, que sirven para corroborar la importancia de las pruebas previas de los GNA.

h) Método de pruebas de **varios generadores eficientes simultáneos** en un mismo proceso de simulación

Se han obtenido resultados muy satisfactorios en el funcionamiento simultáneo pero independiente de varios GNA eficientes probados con la Metodología ya mencionada en la simulación de un sistema (cfr.4.3.2 Pruebas de generadores simultáneos y 5.2.2 Simulación de modelos de precipitaciones de granizo).

El interrogante inicial planteaba:

¿Qué pasa si trabajan simultáneamente en un mismo proceso varios generadores eficientes?

En la citada aplicación, los generadores ineficientes producen valores que el análisis estadístico efectuado en las medias de las variables aleatorias estudiadas, están muy alejados de los rendimientos de los generadores eficientes y de las tendencias históricas del fenómeno natural simulado.

6.2 RECOMENDACIONES

a) De conformidad con la metodología propuesta, (cfr.Figura N°4.3.3), se recomienda el **uso de los generadores congruenciales lineales**, con el grado de confiabilidad que se especifica en cada uno de los siguientes items:

a1) Los GNA más confiables son los que han pasado satisfactoriamente las pruebas estadísticas empíricas(TST01, Chi-cuadrado sobre CS y Kolmogorov-Smirnov sobre KS, primera parte de la metodología), luego las pruebas de entropía continua, y finalmente la prueba teórica espectral:

URN02 Marsaglia y Bray (1968), composición de tres sucesiones (mod 2**32)

URN11 Walker (1967)
$x_{i+1} = 452807053\ x_i \pmod{2**31}$

URN13 Ahrens y Dieter (1974)
$x_{i+1} = 663608941\ x_i \pmod{2**32}$

URN22 Biblioteca MTH$RANDOM de Vax-11/780
$x_{i+1} = 69069\ x_i\ + 1 \pmod{2**32}$

URN35 L'Ecuyer (1987)
$w_{i+1} = 157\ w_i \pmod{32363}$; $y_{i+1} = 146\ y_i \pmod{31727}$;
$z_{i+1} = 142\ z_i \pmod{31657}$; $x_i = (w_i + y_i + z_i - 3) \pmod{32362}$

URN39 Rey (1990)
$x_i = x_{i-1} + (i + x_{i-1})\ sen(i) \pmod 1$

a2) Los GNA que cumplen las mismas condiciones de **a1)** pero que sólo satisfacen la *prueba espectral en alguna de las dimensiones (2, 3, 4)*, son URN01, URN14 y URN30, y su confiabilidad no puede considerarse absoluta.

a3) Los generadores que pasan las pruebas de uniformidad y bondad de ajuste(TST01, opción CS2KS2), pero no las de *entropía continua* ni *la prueba espectral* son URN12 y URN37, quedan clasificados como *sospechosos o ineficientes.*

a4) Sólo a modo de un ejemplo de generadores *ineficientes*, por lo tanto desaconsejables absolutamente para aplicaciones importantes, se citan tres GNA que únicamente superaron las pruebas de entropía: URN06, URN08, y URN09.

Otros generadores, que no pertenecen a Gentstpc pueden ser recomendados si verifican todas las pruebas detalladas en **a1)**, como por ejemplo los GNA de

ciertos paquetes de programas comerciales, estadísticos-matemáticos, analizados en el Cap.5 (cfr. 5.1)

b) No aconsejar el uso de sucesiones de números pseudoaleatorios generados por los **polinomios de Weyl** por no superar las pruebas de la metodología propuesta.

Además, el algoritmo Poliweyl, requiere previamente el cálculo y almacenamiento en memoria de una gran cantidad de cifras fraccionarias de la representación racional que aproxima al número irracional seleccionado.

c) En el empleo de sucesiones de números pseudoaleatorios producidos por generadores de calidad, y utilizadas como archivos-fuente en las aplicaciones a sistemas complejos, se ha observado que **las propiedades de aleatoriedad son más preponderantes, cuanto mayor es la cantidad de valores** utilizados como variables aleatorias, en los respectivos modelos de simulación; por el contrario, las diferencias resultan irrelevantes, cuando se comparan rendimientos con aquellos generadores incorporados a las bibliotecas de los sistemas convencionales, u orientados a procesos de simulación donde las corridas son de reducida longitud.

d) Se recomienda **el empleo de varios GNA eficientes**, cada uno con puntos iniciales(semillas) diferentes por cada flujo de números pseudoaleatorios en un mismo proceso de simulación.

Los GNA de Gentstpc inicializan automáticamente con semillas diferentes cada uno de los lotes de números generados.

e) Se consideran confiables, por la calidad de los generadores incluidos en los paquetes de programas u hojas de cálculo comerciales, aquéllos que han superado las pruebas de Gentstpc, como en los casos, que a modo de ejemplos se analizaron en el Capítulo 5 (cfr. 5.1): NDP Fortran, Matlab, MapleV, Microsoft Office Excel 2003; y no recomendables aquéllos que no las verificaron, como por ejemplo: MS Fortran Visual Workbench v.1.00 y Gauss, v.3.

6.3 SUGERENCIAS PARA POSTERIORES INVESTIGACIONES

a) En el estudio de las pruebas de entropía, un campo interesante lo constituyen los **estimadores de entropía en distribuciones uniformes discretas**, donde una herramienta importante para el desarrollo de algoritmos son las arquitecturas de microcomputadoras superiores a las de 32 bits. Hay ya algunos trabajos sobre este amplio tema.(cfr. Pierre L' Ecuyer, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal)

b) Nuevos generadores del mismo tipo que los estudiados, en particular los GNA **congruenciales lineales multiplicativos** con módulos primos, potencias de dos y mayores que $m = 2^{32}$; y los del tipo **Fibonacci retardados (LFG)**. En ambos casos, el requerimiento de palabras-memoria de 64 o más bits es una condición indispensable, entre las herramientas computacionales.

c) **Generadores no lineales del tipo ICG** (generadores congruenciales de inversión explícita), cuyas estructuras presentan propiedades de interés, pero se necesita incrementar las escasas experiencias prácticas actuales.

d) Incorporación de **otras Pruebas estadísticas** teóricas (v.gr.discrepancia) y empíricas (nuevas pruebas espectrales), en futuras extensiones del sistema Gentstpc.

e) Estudios comparados entre Gentstpc y otras bibliotecas de pruebas estadísticas actuales, en particular las investigaciones de Mersenne (biblioteca Diehard, Florida State Univ.), y Hallekalek (pLAP Project, Univ. Salzburgo)

f) Estudios sobre rangos y períodos de las sucesiones con nuevos GNA, incorporando módulos superiores en los algoritmos congruenciales lineales, y que permitan avanzar en teoría y práctica hacia un modelo para la toma de decisiones que oriente al usuario en la elección del generador más adecuado a su problema.

g) El campo de investigación del **procesamiento en paralelo** para simulación Montecarlo, donde pueden obtenerse sucesiones estadísticamente independientes cuando los números generados en cada procesador lo son, está abierto y en pleno crecimiento.

En particular, la aplicación de la paralelización a métodos de Montecarlo puede estudiarse bajo dos enfoques: la parametrización, donde una familia de

generadores, congruenciales lineales en particular, producen flujos de números pseudoaleatorios que son asignados a los procesadores disponibles en el sistema en paralelo; y el enfoque de partición (splitting), en varias subsucesiones, de las sucesiones de números aleatorios generados, cuyos períodos sean significativamente más extensos que los requeridos por el enfoque de parametrización.

ANEXO I

ANTECEDENTES DE GENTSTPC

En el Cap.4 se hace referencia a algunos antecedentes inmediatos para el diseño de nuestro sistema GENTSTPC. A modo de información complementaria se incluye este capítulo, para proporcionar algunos detalles del sistema Testrand original (1981) y su primera extensión (1986), relativos a los GNA y a las Pruebas Estadísticas Empíricas y Teóricas.

Testrand fue diseñado en 1981 en la Universidad Ohio State por los profesores E. Dudewicz y T. Ralley (Dudewicz y Ralley, 1981) para la generación y pruebas de números pseudoaleatorios.

En 1986, fue readaptado por Z. Karian y I. Mathews en la Universidad de Denison, con otros lenguajes, sistemas operativos y equipos informáticos, que hoy ya también son limitados por su incompatibilidad con equipos y redes de microcomputadores de creciente utilización en diversos medios académicos y tecnológicos del mundo.

En comparación con Testrand, algunos programas fuente existentes URN**, no fueron incluidos en la actual versión GENTSTPC, básicamente por restricciones en su programación para microcomputadores:URN04, URN07, URN15 a URN19, URN23 a URN27.

En la breve descripción que sigue, se destacan las características y restricciones muy particulares de los primeros generadores, que abarcan desde la limitación a un lenguaje de computación de bajo nivel, hasta la programación de algoritmos encriptados.

A.1 GENERADORES ESPECIFICOS DE TESTRAND

El sistema Testrand es una biblioteca de programas que contiene: veinte rutinas para generar, mediante congruencias lineales, sucesiones de números aleatorios uniformemente distribuidos; y catorce pruebas para determinar si una sucesión dada, satisface los criterios de aleatoriedad.

Este paquete consta de veinte generadores, designados con los nombres: URN01, URN02,...,URN20 (Dudewicz y Ralley, 1981, pp. 23-40).

Con posterioridad, en la Denison University (Department of Mathematical Sciences), el proyecto "The generation and testing of pseudo-random numbers on computers" (Mathews y Karian, 1986, pp. 58-79) amplía el sistema Testrand con la incorporación de diez nuevos generadores, URN21 a URN30, elaborados con códigos-fuente de Pascal y Macro (propios de las arquitecturas VAX 11/780, Digital), en reemplazo de Fortran y Assembler (equipos IBM360/370), respectivamente.

URN01

Propuesto por Lewis, Goodman y Miller (1969) y difundido por Learmonth y Lewis (1973) en el paquete denominado LLRANDOM, se trata de un generador con módulo primo $m = 2^{31} - 1$.

$$X_{i+1} = 16807\, X_i \ (\mathrm{mod}\ 2^{31} - 1)$$

Asignar valor a X_0; $X_0 \in (0, 2^{31} - 1)$
Para i = 1, Numale calcular:
$X_i = \mathrm{mod}\ (16807 * X_{i-1}, 2^{31} - 1)$
$U_i = X_i$ / 2147483647
FinPara

URN02

Dado por Marsaglia y Bray (1968). Se trata de un generador del tipo descrito en 2.1.2, es decir de una composición de tres sucesiones:

$$l_i = 65539\, l_{i-1} \ (\mathrm{mod}\ 2^{32})$$
$$m_i = 33554433\, m_{i-1} \ (\mathrm{mod}\ 2^{32})$$
$$k_i = 362436069\, k_{i-1} \ (\mathrm{mod}\ 2^{32})$$

La inicialización utiliza valores enteros impares en un arreglo (NVi) de 128 elementos con números en hexadecimal. El algoritmo se describe brevemente en un pseudocódigo para un lote de Numalea números a ser generados en el vector X_i:

$ml = 65539$
$mm = 33554433$
$mk = 362436069$
$l = 089347405$
$mmm = 301467177$
$kkk = 240420681$
Para i = 1 , $Numalea$, hacer:
$l = l * ml$
$mmm = mmm * mm$
$j = 1 + abs(\ l\ ) * 16777216$
$xi = 0.5 + (\ nv(j) + l + mmm) * 0.23283064E - 9$
$kkk = kkk * mk$
$nv(j) = kkk$
$FinPara$

URN03

Fue propuesto por Marsaglia, Ananthanara Yanan y Paul (1973). La sucesión que define este generador es:

$$x_{i+1} = 69069\ x_i\ (\mathrm{mod}\ 2^{32})$$

Es un generador que para su funcionamiento combina la técnica congruencial y el desplazamiento de bits del tipo F.S.G., que utiliza un desplazamiento a la derecha de 15 y otro de 17 a la izquierda, basándose en el trinomio $x^{17} + x^{15} + 11$. Si X_n e Y_n son los resultados de ambos, respectivamente, expresados en strings de 32 bits, la combinación produce la sucesión Z_i como suma (XOR): $Zi = X_i \oplus Y_i$

URN04

Este generador es el mismo que URN01, pero a cuya sucesión resultante de números pseudoaleatorios se le aplica un método basado en la idea intuitiva de barajadura en un juego de naipes. Consiste en aplicarle una permutación aleatoria (randomizing by shu- ing) como la del algoritmo propuesto por Carter Bays y S. D. Durham (Knuth, 1980, pp. 32-33).

En TESTRAND, ambos subprogramas están confeccionados en el lenguaje Assembler, específicamente para los computadores compatibles con los sistemas IBM/370.

URN05

Es una variante del método de desplazamiento de registros, cfr.2.1.3, de Lewis y Payne (1973) basado en el trinomio primitivo:

$$x^{k} + x^{q} + 1 = x^{89} + x^{38} + 1$$

con m primo igual a $2^{89} - 1$.

En nuestro caso se toma $x^{98} + x^{27} + 1$, con un período de $2^{98} - 1$.

Utilizando la operación XOR en la siguiente fórmula recursiva donde los x_i son los valores de la palabra-memoria del computador, los enteros a generar serán:

$$x_i = x_{i-k+q}\ XOR\ x_{i-k}$$

Para llevar a cabo la codificación, introducimos una tabla de inicialización de 98 números enteros según el algoritmo propuesto por Bratley y otros (op.cit., pp.202-203)

Inicializar X_i ; $i = 1, k$
Asignar: $j = k - q$; $i = k$
Para $i = 1$, $Numalea$
calcular:
$X_i = X_j$ XOR X_i
$j = j - 1$
Si $j = 0$ entonces $j = k$
Si $i = 0$ entonces $i = k$
FinPara

URN06,URN07,URN08 y URN09

URN06 es una versión modificada del generador dado por Marsaglia y Bray (1968). El método corresponde a la clase de métodos compuestos por dos generadores congruenciales multiplicativos de la forma:

$$\begin{aligned} v_{i+1} &= 65539\ v_i \pmod{2^{31}} \\ w_{i+1} &= 262147\ w_i \pmod{2^{32}} \\ x_{i+1} &= (v_{i+1} + w_{i+1}) \pmod{2^{31}} \end{aligned}$$

Como se justifica por razones de hardware, IBM 360/370, (Dudewicz y Karian, p.12), el módulo del generador compuesto es $m = 2^{31}$, para la sucesión suma $(\mathrm{v}_{i+1} + \mathrm{w}_{i+1})$.

El código de computación es simple, a partir del pseudocódigo:

Asignar Valores Iniciales:
$V_0 = 524287$
$W_0 = 654345465$
Para i = 1, $Numalea$
calcular:
$V_i = V_{i-1} * 65539$
$W_i = W_{i-1} * 262147$
$X_i = 0.4656612873 * 10 * *(-9) * abs\ (V_i + W_i)$
FinPara

En el caso de los generadores URN07,URN08 y URN09 se utilizará:

$$x_{i+1} = 65539\ x_i\ (\mathrm{mod}\ 2^{31})$$

El subprograma URN07 de TESTRAND es específico para las máquinas IBM 360/370 y está escrito en lenguaje Assembler, en cambio URN08 lo está en FORTRAN para el sistema SSP de IBM, siendo levemente inferior en velocidad que la versión en Assembler.

El subprograma URN09 es similar a URN08, fue escrito como la versión de Nie, Bent y Hull (1973) incluida en el paquete estadístico SPSS. Los tres generadores son versiones no muy diferenciadas del generador RANDU de IBM (1970).

Utilizando aritmética de "complemento a dos" (Bratley, Fox y Schrage, 1987, pp. 211-212) se puede ejecutar el siguiente algoritmo en equipos PC:

Asignar:
Valor a $X_0 \in (0, 2^{31} - 1)$
$X_m = 2147483648$; $X_{m1} = 2^{-31}$
Para i =1, $Numalea$
calcular:
$X_i = X_0 * 65539$
Si $X_i < 0$ hacer:
$X_i = 1 + (X_i + 2147483647)$

FinSi
$Xo = X_i$
$U_i = X_i * X_{m1}$
FinPara

URN10

Es otro generador multiplicativo congruencial dado por Kruskal (1969).

$$x_{i+1} = 5^3 x_i \pmod{2^{13}}$$

Posee la propiedad de ser usable en distintos microcomputadores debido al requerimiento de la palabra-memoria, lo que lo hace muy portable. Sin embargo, su período de $2^{11} = 2048$ es adecuado sólo para usos que requieran una cantidad reducida de números pseudoaleatorios.

Pseudocódigo para URN10:

Asignar: $X_0 = 1$
$m = 8192$
$flm = m$
Para j=1, $Numalea$
calcular:
$K = Xo$
Para i = 1,3
calcular:
$K = \text{mod}\ (5 * K, m)$
FinPara

$Xj = K/flm$
$iy = K$
$X_0 = iy$
FinPara

URN11

Es un generador congruencial mixto:

$$x_{i+1} = a x_i + c \pmod{2^{31}}$$

Walker (1967) consideraba: $a = 452807053,\ \ c = 0.$
En TESTRAND: $a = 5^{15} = 30517578125$
Asignar: $X_0 = 1;\ l = 452807053\ ;\ zz = 2^{31}$
Para i $= 1,\ Numalea$
hacer:
$k = X_0$
$z = l * k + c$
$z = mod\ (z, zz)$
$z = abs\ (z)$
$k = z$
$X_i = z\ /\ zz$
$iy = k$
$X_0 = iy$
FinPara

URN12

Es la versión de URN11 en doble precisión como lo tiene incorporado el paquete TESTRAND.

URN13

Debido a Ahrens y Dieter (1974) el generador congruencial multiplicativo es:

$$x_{i+1} = 663608941\ x_i\ (\mathrm{mod}\ 2^{32})$$

La programación fue realizada en Assembler para el sistema IBM/370.

En 4.4.1.2, se esquematiza el modelo computacional que en la versión GENTSTPC se ha diseñado, para evitar problemas de desbordamiento, debido a que el módulo de este generador congruencial es superior a $2^{31} - 1$.

La misma idea fue aplicada a URN22 también con resultados muy satisfactorios ya que en la versión precedente de Z. Karian (Denison University) el algoritmo no había sido aún codificado en un lenguaje de alto nivel compatible con microcomputadores.

URN14

Es un generador congruencial mixto creado por Zarling (1971). Utiliza inicialmente una sucesión:

$$x_{i+1} = 266245x_i + 453816693 \,(\mathrm{mod}\ 2^{31})$$

Luego, en base a una tabla N_i, para $i = 1, 64$, realiza una permutación aleatoria. Los valores en esa tabla son seleccionados empleando otra sucesión $\{Z_i\}$ definida así:

$$z_{i+1} = 10924\ z_i + 6925 \qquad [1]$$

$$z_{i+1} = z_{i+1} - 32769\frac{z_{i+1}}{32769} \qquad [2]$$

El pseudocódigo propuesto en GENTSTPC para su programación en equipos PC es:

Ingresar tabla de números aleatorios N_i ; $i = 1, 64$
$iy = 197249728$; $m1 = 0.4656612873 * 10^{-9}$; $iz = 0$
Para i = 1, $Numalea$
calcular:
$iy = 266245 * iy + 453816693$
Si $iy < 0$ entonces:
$iy = iy + 2147483647 + 1$
FinSi

$iz = 10924 * iz + 6925$
$iz = iz - 32769 * (iz/32769)$
Si $iz = 32768$ entonces:
Calcular iz según [1]
FinSi
$ind = (iz\ /\ 512) + 1$
$Xi = m1 * Nind$
$Nind = iy$
FinPara

URN15, URN16, URN17, URN18, URN19 y URN20

Estos seis generadores congruenciales multiplicativos tienen el mismo módulo $m = 2^{31} - 1$. Se deben a D.C.Hoaglin (1976).

Los factores multiplicativos son:

$a15 = 764261123$
$a16 = 1323257245$
$a17 = 1078318381$
$a18 = 1203248318$
$a19 = 397204094$
$a20 = 2027812808$

En TESTRAND están empleados específicamente en programas fuente Assembler IBM/360/370.

URN21

Se debe a C.G. Swain y M.S. Swain (1979). Se define mediante cuatro sucesiones:

$$
\begin{aligned}
&x_{i+1} = x_i + x_{i-1} + x_{i-2} \\
&x_{i+1} = x_{i+1} + 1357 \qquad si \;\; x_{i-1} < 50000000 \\
&x_{i+1} = x_{i+1} - 100000000 \;\; si \;\; x_{i+1} \geq 100000000 \\
&x_{i+1} = x_{i+1} - 100000000 \;\; si \;\; x_{i+1} \geq 100000000
\end{aligned}
$$

El algoritmo puede expresarse así:

Asignación de semillas: $m1 = 32007779$; $m2 = 23717810$; $m3 = 52636370$
Para i $= 1$, $Numalea$
hacer:
$m4 = m1 + m2 + m3$
si $m2 <$ 50000000 entonces
  $m4 = m4 + 1357$
si $m4 \geqslant 100000000$ entonces
 $m4 = m4 - 10000000$
si $m4 \geqslant 10000000$ entonces
  $m4 = m4 - 10000000$
FinSi
$m1 = m2$; $m2 = m3$; $m3 = m4$
$x\ i = m3 * 10^{-8}$

FinPara

URN22

Este generador pertenece a la biblioteca MTH$RANDOM de rutinas de los computadores VAX-11/780 bajo el sistema operativo VMS. En particular, el lenguaje GPSS-VX utiliza los números pseudoaleatorios generados por ese sistema.

$$x_{i+1} = 69069\ x_i + 1 \quad (\text{mod } 2^{32})$$

Salvo por el valor de la constante aditiva C=1, este generador con C = 0 es el URN03.

URN23

Se trata de un generador congruencial mixto propuesto por T.A. Brody (1984) y referido al método de Mc Laren - Marsaglia, que emplea dos generadores congruenciales y una tabla de números aleatorios.

La sucesión:

$$x_{i+1} = 69069\ x_i + 1 \quad (\text{mod } 2^{32})$$

es utilizada para generar x_{i+1}, conociendo x_i. Otra sucesión es creada mediante la operación lógica XOR de x_{i+1} y una de las ocho componentes de la tabla.

El esquema del algoritmo es:

$x_{i+1} = 69069 * x_{i+1} \quad (\text{mod } 2^{32})$
$v_{i+1} = x_i\ XOR\ T_{j*i}$
$j = Bits\ 13..15\ de\ T_{j*i}$; el próximo índice tabla T
$T_{j*i} = v_i$
$u_{i+1} = v_{i+1} / 2^{32}$

La semilla de $\{x_i\}$ es 65539 y los valores hexadecimales de la tabla T: 347A50E5; 326C0AF7; 0DE685A8; 769727F9; 1BBA2C16; 10BABA9C; 11942E23 y 39CC478E.

URN24

Este generador está orientado a microcomputadores específicamente, donde hasta hace poco tiempo, el rango de la palabra-memoria era 16 bits. A. Thesen

(1985) sugirió un algoritmo que en vez de producir una sucesión de números cualesquiera y después convertirlos al intervalo (0,1), directamente genera la mantisa y luego el exponente.

De tal manera que se emplean dos generadores congruenciales diferentes. Ambos se basan en palabras de 16 bits de modo que los números producidos están en el intervalo (0, 2^{16} - 1).

Los números generados $\{u_i\}$ en (0,1) empleando un lenguaje de bajo nivel utiliza dos semillas (que son enteros entre 0 y 2^{16} -1); y dos constantes multiplicativas (MULT), para los bytes generadores, de modo que el generador congruencial mixto se expresa:

$$u_{i+1} = MULT * u_i + 1 \pmod{2^{15}}$$

El algoritmo tiene en cuenta particularidades de la arquitectura del procesador por lo que habrá que tener presente las especificaciones del equipo con el que se trabaje para adaptarlo convenientemente.

Los estándares referidos a la representación en punto flotante vigentes desde 1985(ANSI/IEEE Std 754-1985) consideran números x ($\neq$0) normalizados, es decir:

$$x = m * 2^E \; con \; 1 \leq \; m \; < 2$$

lo que significa que la representación de la mantisa en base binaria es:

m = $\pm \; (b_0 b_1 b_2 b_3 ...)_2$, con $b_0 = 1$ (bit oculto)

La parte fraccionaria es dada por el string $b_1 b_2 b_3 ...$

El IEEE (Institute for Electrical and Electronics Engineers) prevé tres tipos estándar de punto aritmético: simple precisión, doble precisión y precisión extendida; los dos primeros requieren palabras de 32 y 64 bits. Ilustración de la representación en IEEE de una palabra-memoria en simple precisión:

$$\pm \quad a_1 a_2 ... a_8 \quad b_1 b_2 ... b_{22} b_{23}$$

Ejemplo:

Si el string de 8 bits del exponente fuese $(11111100)_2 = 252_{(10}$, el valor numérico representado sería $\pm \; (1.b_1 b_2 ... b_{23} \;) \times 2^{125}$, donde el exponente 125 se obtuvo después de restarle el "exponente sesgado" de valor E= 127.

Los valores mínimo y máximo corresponden a:

$(1.00...0)_2 * 2^{-126}$ y $(1.11...1)_{(2} * 2^{127}$

los cuales en base decimal dan aproximadamente

$1.175 * 10^{-38}$ y $3.403 * 10^{38}$

En uno de los compiladores utilizado en GENTSTPC, NDP Fortran (NDP Fortran Microway, 1993, pp. 16-17), los números reales de simple precisión ocupan 4 bytes de memoria, con una precisión entre 6 y 7 dígitos decimales.

En doble precisión (REAL * 8) el rango, en valor absoluto, es de $1.8 * 10^{-308}$ a $2.23 * 10^{308}$.

URN25

Este generador se debe a J. Gason, (1977) y propone el uso de un algoritmo de encriptado estándar (DES).

Es de carácter muy complejo y se lo codifica en lenguajes de bajo nivel. Esencialmente forma permutaciones con los bits y combina sucesiones de ellos mediante la operación lógica XOR.

URN26

Se debe a Mathews y Karian (1986) con el propósito de mejorar el URN24, pero los resultados obtenidos fueron desalentadores.

URN27

Es un generador de tipo descrito en 2.1.3, FSR (Feedback Shift Register) o de Tausworthe, incluido en el Lenguaje de Simulación GPSS/ H (1983), que se basa en el trinomio:

$x^{31} + x^3 + 1$

Según Whittlessey este generador tiene un período de 2^{31} - 1 y propone el siguiente algoritmo, donde R1 y R2 son registros :

R1 = SEED
R2 = LOGICAL SHIFT RIGHT 3 (R1)
R2 = R2 XOR R1
R1 = R2
R2 = LOGICAL SHIFT LEFT 28 (R2)
R1 = R1 XOR R2
R1 = ABS (R1)
RND = R1 / 2** 31

URN28

Es un generador del mismo tipo que URN01 donde el factor multiplicativo es 16807 y el módulo(2^{31} - 1).

Su programación se hace en Assembler para obtener mayor velocidad.

Este algoritmo sugerido por Lewis y Lehmer fue discutido por L. Schrage, con el propósito de ser codificado en un lenguaje de alto nivel para lograr mayor portabilidad.

Puesto que el módulo no es potencia de 2 los productos $16807x_i$ pueden producir desbordamiento (overflow) de las palabras-memoria de 32 bits.

Para evitar esos efectos, Schrage propone la descomposición de todo número de 32 bits en la forma $(\alpha * 2^{16} + \beta)$ donde α y β tienen respectivamente 15 y 16 bits.

Expresión del generador :

$$x_{i+1} = 16807 x_i \pmod{2^{31} - 1}$$

de donde, al sustituir, resulta:

$x_{i+1} = (16807 * \alpha) * 2^{16} + (16807 * \beta) \pmod{2^{31} - 1}$

Para realizar la complicada operación de módulo, Schrage sigue el procedimiento de descomponer el cálculo de la congruencia:

$x_{i+1} = a * x_i \pmod{p}$ mediante la congruencia
$z = a * x_i \pmod{m}$, que se supone fácil de calcular y tal que:
$p = 2^{n-1} = 2^{31-1}$; m = $2^n = 2^{31}$

Entonces :

$z = a * x_i - (k * 2^{31})$; k = parte entera de $(a * x_i \ / \ 2^{31})$

Sumando k en la relación anterior :

$z+k = a * x_i - k * (-1 + 2^{31}) = a * x_i - k * p$

Como $2 + k < 2 * (2^n - 1) = 2 * p$

por lo tanto:

$x_{i+1} = z + k$ *si* $z + k < p$
$x_{i+1} = z + k - p$ *si* $z + k > p$

Matheus y Karian dan una versión en Lenguaje Pascal de este programa.

URN29

$$x_{i+1} = 16807 x_i \pmod{2^{31} - 1}$$

Es una versión muy particular de URN01 que D. Schmidt utilizó en 1986 en un trabajo en colaboración con Matheus y Karian referido a integración por Monte Carlo para el cálculo de integrales triples en una aplicación de física nuclear. La única diferencia con URN01 es que para obtener la sucesión $U_i = x_i \ / \ (2^{31} - 1)$, divide por un número mayor: $(2^{31} - 1) + 2^6$, lo que no permite obtener números próximos a 1 en el intervalo (0,1).

URN30

Es el generador de Lehmer, una de cuyas implementaciones se debe a Payne, Rabung y Bogyo (1969)

$$x_{i+1} = 6303600016\, x_i \pmod{2^{31} - 1}$$

Es utilizado por el lenguaje de simulación SIMSCRIPT II.5 y su codificación ha sido realizada en Assembler, Fortran, C y Pascal, teniendo en cuenta las dificultades de cálculo en relación con el módulo, lo que ya se ha explicado para otro generador análogo como el URN28.

En Law y Kelton (A. M. Law y W.D. Kelton, 1991, pp. 449 -456) se desarrolla el código Fortran, de Marse y Roberts (1983), con los resultados de su programación en distintos microcomputadores y compiladores, así como también en grandes equipos como CRAY y DEC.

A.2 PRUEBAS EMPIRICAS EN TESTRAND

La biblioteca de pruebas contiene once pruebas empíricas clásicas donde cada una produce los estadísticos Chi-cuadrado y Kolmogorov-Smirnov, además de tres pruebas teóricas.

En una corrida de prueba donde, por ejemplo se generan 1'000 lotes de 10'000 números cada uno, el sistema permite verificar la *uniformidad* de la distribución y la *bondad de ajuste* para lo cual produce los estadísticos χ^2 y K-S para cada lote, respectivamente. Tales estadísticos - 1'000 - son almacenados y luego probados para comparar sus distribuciones empíricas y teóricas.

A.2.1 Recolector de cupones

Esta prueba, llamada también del coleccionista de cupones, y designada TST04 produce un estadístico que, bajo hipótesis de aleatoriedad de la sucesión, verifica si $\{U_i\}$ tiene una distribución Chi-cuadrado con un número de grados de libertad, previamente determinado.

Dada una sucesión $\{U_i\}$ de números en (0,1), se los convierte en enteros 1,2, ... ,D, mediante las relaciones:

$$U_j \text{ se sustituye por } 1 \text{ si } 0 \leq U_j < \frac{1}{D}$$

$$U_j \text{ se sustituye por } 2 \text{ si } \frac{1}{D} \leq U_j < \frac{2}{D}$$

. . . . . .

$$U_j \text{ se sustituye por } D \text{ si } \frac{D-1}{D} \leq U_j \leq 1$$

Entonces, en la nueva sucesión serán "aleatorios" de 1 a D, siempre que los $\{U_j\}$ lo sean.

La prueba consiste en tener representados todos los enteros desde 1 a D. Si se elige el entero 3 deberán aparecer 3 números U_j que verifiquen:

$$\frac{2}{D} \leq U_j < \frac{3}{D}$$

Sea Q el número de términos de la sucesión que deben ser examinados para hallar un conjunto completo de D números enteros.

Entonces, los valores posibles de Q son: Q = D, Q = D+1, Q = D+2, ...

Cuando se supone hipótesis de aleatoriedad se conocen las respectivas probabilidades:

P(Q=D), P(Q=D+1), P(Q=D+2), ...

Calculando repetidamente Q entonces se puede realizar una prueba Chi-cuadrado para constatar si los números observados en cada categoría concuerdan con las probabilidades teóricas.

La prueba se clasifica en 3 casos:

1. CUPON D = 5

Es aplicable a los U_i. Las categorías son: Q = 5, Q = 6, ... ,Q = 19, Q $\geqslant$ 20

2. CUPON D = 5, FR(100*U)

Es similar al anterior, pero aplicado a la parte fraccionaria de (100*U_i)

3. CUPON D = 10

Es aplicado a los U_i. Las categorías son 6.

10 $\leq$Q $\leq$ 19 ; 20 $\leq$ Q $\leq$ 23 ;
24 $\leq$ Q $\leq$ 27 ; 28 $\leq$ Q $\leq$ 32 ;
33 $\leq$ Q $\leq$39 ; Q $\geqslant$ 40

Para la realización de la prueba, cada una de las categorías en los tres casos tiene asignada una probabilidad teórica, en total 22 números en el intervalo (0,1).

Las salidas de la prueba para cada uno de los 3 casos dan información acerca de la longitud del segmento, el número esperado y observado, el valor de Chi-cuadrado con una cierta cantidad de grados de libertad y una tabla de las probabilidades para que una variable aleatoria χ^2 sea menor o igual que x, el valor crítico de la tabla.

A.2.2 Prueba del intervalo o de huecos (GAP)

Designada como TST05, consiste en examinar la lista de los números pseudoaleatorios $\{U_i\}$ para ver si se encuentran o no, entre α y β , $\alpha < \beta$.

Cada número encontrado es reemplazado por un "1", y en caso contrario por un "0". La nueva sucesión que reemplaza a $\{U_i\}$ es la formada por "unos" y "ceros".

Sea $p = \beta$ - α. Si los números dados fuesen realmente aleatorios, entonces la probabilidad de que ocurran "j" ceros después de un "1", y antes de la ocurrencia de otro "1", o probabilidad de longitud "j", se define así:

$$p_j = p(1-p)^{\,j} \quad \text{con} \quad j = 0, 1, ..., 9$$

Se observan las frecuencias con las que ocurren las "j" categorías y luego se realiza una prueba Chi-cuadrado para ver si las frecuencias observadas en cada categoría coinciden con las probabilidades teóricas.

El procedimiento de Kendall-Babington opera con $\alpha = 0$ y $\beta = 1$.

Los casos particulares son:

$(\alpha\ , \beta\ ) = (0, 0.5)$	se llama GAP por debajo de la media
$(\alpha\ , \beta\ ) = (0.5, 1.0)$	se llama GAP por encima de la media
$(\alpha\ , \beta\ ) = (0.333, 0.667)$	se llama GAP de los dos tercios.

A.2.3 Prueba de permutación

Denominada TST07. Se agrupan los números de la sucesión $\{U_i\}$ en t-uplas:

$$u_{it}, u_{it+1}, u_{it+2}, ..., u_{(i+1)t-1}, \quad i \geq 0$$

Cada uno de esos conjuntos de T números tiene T! posibles ordenamientos. Bajo hipótesis de independencia cada uno de los ordenamientos tiene una probabilidad $\frac{1}{T!}$ (todas las permutaciones son equiprobables).

Partiendo de las T! categorías ordenadas se observa cuántas de las t-uplas formadas a partir de t = 3,4,5 caen en cada una de esas categorías.

Luego se aplica Chi-cuadrado para determinar si los números observados en cada categoría están de acuerdo con las probabilidades asignadas a cada categoría.

Ejemplo: Permutación de t = 3 objetos

Salidas: Cantidad de grupos de 3 números examinados; frecuencia esperada de cada tipo de permutación; valor del estadístico χ^2_5

Permutaciones: (123) (132) (321) (213) (231) (312)

Ocurrencias: 572 541 548 566 533 573

El programa da resultados separadamente para permutaciones de 3, permutaciones de 4 y permutaciones de 5 elementos.

A.2.4 Prueba del póker

Se designa TST08. La lista de números $\{U_i\}$ se convierte en números enteros 1,2,...,10, con los que se conformará otra lista de números pseudoaleatorios. El método abarca "manos" de 4, con 4 reparticiones; "manos" de 5, con 6 reparticiones; y "manos" de 5, con 4 reparticiones.

Luego se toman sucesivos conjuntos de 5 enteros, y en cada uno se determina si contienen el mismo número AAAAA, siendo A cualquier cifra de 1 a 10; luego si contiene 4 cifras iguales y una distinta, AAAAB; 3 iguales entre sí, y 2 iguales entre sí, AAABB; 3 iguales y 2 distintas, AAABC; 2 pares distintos pero iguales entre sí en cada par, AABBC; un par igual pero los tres restantes diferentes entre sí, AABCD; o finalmente las 5 cifras diferentes.

Son 7 reparticiones cuyas probabilidades de ocurrencia, bajo hipótesis de aleatoriedad son conocidas:

P (AAAAA) = 0.0001 ; P (AAAAB) = 0.0045 ; P (AAABB) = 0.0090 ;
P (AAABC) = 0.0720 ; P (AABBC) = 0.1080 ;
P (AABCD) = 0.5040 ; P (ABCDE) = 0.3024

Las reparticiones de 7 pueden combinarse de modo de tener "manos" de 5 con reparticiones de 5 y 6. La prueba contempla un caso de "manos" de 4, con 4 reparticiones.

En las salidas se brinda información acerca de la cantidad de grupos de 4 ó 5 "manos", con especificación del tipo de repartición, número de casos observados y esperados.

La prueba χ^2 se aplica para analizar si los números observados en cada repartición concuerdan con las probabilidades teóricas.

Se toman luego sucesivos pares de números de la nueva sucesión para determinar a cuál de las M categorías pertenecen.

Bajo hipótesis de aleatoriedad las categorías son equiparables, cada una con una probabilidad igual a $1/M^2$, siendo $M = 3$, $M = 10$, y $M = 20$.

En las salidas se muestran la cantidad de pares (Q, R) de números aleatorios probados con Q y R entre 1 y 3 ($M = 3$); o entre 1 y 10 ($M = 10$); o entre 1 y 20 ($M = 20$); el número esperado en cada categoría y los respectivos estadísticos Chi-cuadrado con 8 grados de libertad.

A.2.7 Prueba del producto rezagado

Se incluye en el paquete como TST03 (Lagged Correlation Test).

Dada la sucesión $\{x_1, x_2, ..., x_n\}$, para probar que no existe correlación entre x_i y x_{i+K} se define un coeficiente C_K, llamado coeficiente del producto rezagado de la sucesión; y donde K es la longitud del rezago.

$$C_K = \frac{1}{N-K} \sum_{i=1}^{N-K} x_i x_{i+K}$$

Si no existe correlación entre x_i y x_{i+K} los valores de C_K tenderán a distribuirse normalmente.

En esta prueba se espera que el coeficiente LAG "K" de correlación sea pequeño.

Se acepta que el 95% de las veces se puede encontrar un valor K entre

$$-(2\sqrt{N}+1)/N \quad \text{y} \quad (2\sqrt{N}-1)/N$$

Fuera de este intervalo el generador es sospechoso.

Hay 4 métodos conocidos para calcular el coeficiente de correlación, cuyos resultados se muestran en las salidas del programa de computación, siendo la última columna la que resume, afirmativamente o no, la aleatoriedad para distintas longitudes del rezago (K=1,20).

A.2.8 Prueba de los máximos

Se designa como TST11. Dada una sucesión $\{U_i\}$ de N números pseudoaleatorios, se forman sucesivos conjuntos de t números cada uno, y luego se determina el máximo de cada conjunto.

Bajo hipótesis de aleatoriedad de los números originales, la distribución de la sucesión de números máximos es conocida.

La prueba Kolmogorov-Smirnov evalúa la desviación entre dicha distribución y la función de distribución "empírica" de la sucesión de máximos.

Las medidas a utilizar, si se tienen M conjuntos de t números cada uno, son:

$$K^+ = \sqrt{M}\ max\{Empírica(x) - x^t\}$$
$$K^- = \sqrt{M}\ max\ \{\ x^t - \ \ Empírica(x)\}$$
$$K^* = max\ \{K^+, K^-\}\ , \qquad x \in [0,1]$$

Los valores de t permitidos son t = 2,3,4,...

A.3 PRUEBAS TEORICAS

Son también llamadas "tests a priori" porque no requieren la generación de las sucesiones $\{U_i\}$ sino que, referidas por ejemplo a los generadores congruenciales mixtos, basta analizar el *módulo* y las *constantes* multiplicativa y aditiva, para conocer el comportamiento de los términos de la sucesión en su período completo.

A.3.1 Prueba de correlación serial

Se incluye en el paquete como TSTM1 y es aplicable a generadores congruenciales mixtos.

Si bien la prueba TST03 puede también calcular la correlación serial empíricamente, el TSTM1 determina en forma directa mediante fórmulas teóricas cuál es la correlación más próxima a cero (LAG1).

En este caso el valor del factor multiplicativo debería ser elegido próximo a $\sqrt{m}$.

Según Karian y Dudewicz (Karian y Dudewicz, 1991, p.105) se puede demostrar que si $\{U_i\}$ está formada por números aleatorios, el coeficiente de correlación es igual a 0.

De la expresión analítica se deduce una cota superior del error en función del factor multiplicativo a y del módulo m, es decir: $abs(\square) \leq (a+6)/m$.

Si mcd(a,m) =1, el valor del error puede ser reemplazado por un valor exacto, como es el caso del generador URN13.

A.3.2 Prueba de las distancias interplanares

Llamado también prueba espectral. Se encuentra en Testrand con la denominación TSTM4.

Está basado en trabajos de Marsaglia (1968) que afirmaba que los números aleatorios se disponen principalmente sobre planos.

El nombre "espectral" fue propuesto por Coveyou y Mac Pherson (1967).

Básicamente esta prueba proporciona una medida de la uniformidad en una dimensión N-ésima para los números generados por un algoritmo del tipo:

$$x_{i+1} = a\ x_i + c\ (\mathrm{mod}\ m)$$

en un ciclo completo.

Más específicamente, se refiere a las d-uplas formadas por los U_i que pueden llenar el hipercubo unitario [0,1]^d.

Marsaglia encontró una cota de los hiperplanos que pueden ser cubiertos por todas las d-uplas. La prueba considera la máxima distancia entre todas las familias de hiperplanos paralelos.

En aquellos casos donde el número de hiperplanos en una familia dada es muy pequeño, se afecta la aleatoriedad de la sucesión en su período completo.

Si dos de esos hiperplanos están muy separados, entonces el generador deja grandes huecos en esos espacios, lo que indica una baja calidad del generador en el espacio al que pertenecen las d-uplas.

El generador congruencial genera la sucesión $\{U_i\}$ de números con los que, fijado un espacio n-dimensional, podemos obtener las n-uplas

$$(U_1, ..., U_n)\,;(U_2, ..., U_{n+1})\,;(U_3, ..., U_{n+2})\,;...$$

que pertenecen a una pequeña cantidad finita de hiperplanos paralelos igualmente separados.

Ese pequeño número es a lo sumo $(\sqrt{n}.m)^{\frac{1}{n}}$ como lo propuso Marsaglia.
Ejemplo: para $m = 216$ y $n = 2$, da aproximadamente 304.

La prueba espectral da una indicación de la calidad del generador en función de la distancia máxima d_n entre hiperplanos paralelos, tomada sobre todas las familias de hiperplanos paralelos que cubren todos los puntos de una sucesión.

Sea el generador:

$$x_{i+1} = 7x_i \pmod{11}, \quad x_0 = 1$$

La prueba espectral para $n = 2$ provee la sucesión:
$\{x_i\} = \{1, 7, 5, 2, 3, 10, 4, 6, 9, 8, 1, ...\}$
de período $(m - 1) = 10$.
Los pares (u_i, u_{i+1}) obtenidos después de dividir $x_i\,/\,11$ son :
$(1, 7); (7, 5); (5, 2); (2, 3); (3, 10); (10, 4); (4, 6);$
$(6, 9); (9, 8); (8, 1)$

los que representados en el espacio R^2 muestran a los 10 puntos pertenecientes a distintos hiperplanos (rectas) paralelos equidistantes.

Para n fijo, el mejor generador producirá una distancia d_n que sea mínima.

El criterio propuesto es que $d_n \leq 2^{\frac{-30}{n}}$, para n = 2,3 ...,9, aunque la prueba se realiza para valores entre 2 y 6.

Las dimensiones 2, 3 y 4 parecen ser más útiles para detectar generadores deficientes.

La "mejor" distancia interplanar calculada en función de n y el módulo m del generador puede expresarse como:

$$b_m = 1/\left[(n!m)^{\frac{1}{n}} - 1\right]$$

La relación de esta fórmula con d_n se puede dar como el cociente:

$$r = d_n/b_m$$

que alcanza 1, como valor mínimo; aquellos valores próximos a 1 son los mejores.

Para calcular la distancia d_n si $c \neq 0$, y período m, ó $c = 0$ y período $(m-1)$, se ha diseñado un algoritmo en el subprograma TSTM4 (GENTSTPC), que es independiente del tamaño de la palabra-memoria del computador.

Un desarrollo reciente acerca de esta prueba espectral fue realizado en la Universidad Nacional del Comahue con sistemas operativos y lenguajes diferentes a los empleados en GENTSTPC. *(http://www2.uncoma.edu.ar/~oso/spectral)*

Una versión anterior a la mencionada fue diseñada por Z. Karian en un lenguaje simbólico (Denison University).

REFERENCIAS BIBLIOGRAFICAS

[1] Admirat P., *Étude expérimental du phénomene grêle et de sa modification par la technique soviétique*, Groupement National d'Etudes des Fléaux Atmosphériques, Grenoble, 1982.

[2] Altaber J.L., *Représentations arithmétiques de grandeurs aléatoires*, Tesis, Faculté des Sciences, L 'Université de Clermont, Clermont-Ferrand, 1967.

[3] Banks J., Carson J.S., y Nelson B.L., *Discret-Event System Simulation*, 2da. ed., Prentice-Hall, New Jersey, 1996.

[4] Batanero C., *Aleatoriedad, Modelización, Simulación*, X Jornadas sobre el Aprendizaje y la Enseñanza de las Matemáticas, Zaragoza, 2001

[5] Bernhofen L., Dudewicz E., Levendovszky J. y Van der Meulen E., *Ranking of the best random number generators via entropy-uniformity theory*, American Journal of Mathematical and Management Sciences, v.16, 1996.

[6] Bratley P., Fox B.L. y Schrage L.E., *A Guide to Simulation*, 2da. ed., Springer, New York, 1983.

[7] Brody, T. A., *A Random-Number Generator*, Computer Physics Communications, pp. 39-46, 1984.

[8] Carrasco Gi , A., *Empleo Operativo del UD-13*, Revista de Aeronáutica y Astronáutica, Agosto 1989, Madrid.

[9] Chaitin G. J. ,*Randomness and Mathematical Proof*, Scientific American, May 1975.

[10] Char B. y otros, *Maple V Language Reference Manual*, Springer-Verlag, 1991.

[11] Char B. y otros , *A tutorial Introduction to Maple V*, Springer-Verlag, 1992.

[12] Chen Bangtian, McCoskey Suzanne y Kao Chihwa, *Estimation and Inference of a Cointegrated Regression in Panel Data: A Montecarlo Study*, American Journal of Mathematical and Management Sciences, v.19, 1999.

[13] Dudewicz Edward, Karian Zaven y Marshall Rudolph, *Random Number*

Generation on Microcomputers, Modeling and Simulation on Microcomputers, The Society for Computer Simulation, Ed. R. Greer Lavery, La Jolla, California, 1985.

[14] Dudewicz, Edward y Ralley Thomas, *The Handbook of Random Number Generation and Testing with TESTRAND Computer Code*, American Series in Mathematical and Management Sciences, American Sciences Press, 1981.

[15] Dudewicz Edward y Karian Zaven, *Modern Desing and Analysis of Discrete Event Computer Simulations*, IEEE Computer Society Press, 1985.

[16] Dudewicz E. J. y Mishra S., *Modern Mathematical Statistics,* J. Wiley, N. York, 1988.

[17] Figuera Andú J., *PERT-CPM-ROY*, Ed. Autor S.A.E.T.A., Madrid, 1966.

[18] Jansson Birger, *Random Number Generators*, Victor Pettersons Bokindustriab, Stochholm, 1966.

[19] Fishman G. S., *Discrete-Event Simulation*, Springer-Verlag, New York, 2001

[20] Fishman George S., *Conceptos y métodos en la simulación digital de eventos discretos,* Limusa, México, 1978.

[21] François Charles, *Diccionario de Teoría General de Sistemas y Cibernética,* Asociación Argentina de Teoría General de Sistemas y Cibernética, División Argentina de la International Society for the Systems Sciences, Buenos Aires, 1992.

[22] GPSS/PC, *Reference Manual*, Minuteman Soft, Massachusetts, 1988.

[23] GPSS World, *Reference Manual*, 2000 Minuteman Software, Massachusetts.

[24] Heal K.M. et al., *MAPLE V Learning Guide, Waterloo Maple Inc.,* 1996.

[25] Hellekalek P., *Good random number generators are (not so) easy to find*, Salzburg University, Mathematics and Computers in Simulation 46 (1998),485-505

[26] Hoaglin D.C., *Theoretical properties of congruential random-number generators: an empirical view,* Department of Statistics, Harvard University, Cambridge,MA, 1976.

[27] IBM, *General Purpose Simulation System V, Introductory User's Manual,* IBM Corporation, White Plains, N.York, 1971.

[28] Jennergren L.P., *Another method for random number generation on microcomputers,* Simulation 40, 79, 1984.

[29] Karian Z. A. y Dudewicz E. J., *Modern Statistical, Systems, and GPSS Simulation,* Computer Science Press, W.H. Freeman, N. York, 1991.

[30] Keefer D.L., Verdini W.A., *Better Estimation of PERT Activity Time Parameters*, Management Science, vol.39, N° 9, 1993.

[31] Kerrigan James F., *Migrating to Fortran 90,* O'Reilly Associates Inc., USA, 1993.

[32] Kleijnen Jack P.C. y Van Groenendaal J.H., *Simulation: A Statistical Perspective,* Dept.of Business and Economics, Katholieke Universiteit Brabant, Tilburg, Netherlands, 1991.

[33] Knuth ,Donald , *The Art of Computer Programming, Seminumerical Algorithms*, vol.2, Addison-Wesley, 1981.

[34] Kruskal J.B., *Extremeley portable random number generator,* Communications of the ACM, 12, 1969.

[35] Lassig J., Ruiz J., Olmos A. y otros, *Evaluación económica de los daños históricos por granizo en el Alto Valle de Río Negro y Neuquén*, U.N.C., 1987.

[36] Lassig, J., *Especificaciones para el Diseño de un Hidroavión bombero pequeño para las provincias de Neuquén, Río Negro y Chubut,* Taller de Trabajo en Incendios Forestales, San Martín de los Andes, UNC, 1989.

[37] Law Averill y Kelton W. David , *Simulation Modeling & Analysis*, McGraw - Hill, 1991.

[38] L 'Ecuyer P. , *"Software for Uniform Random Number Generation: Distinguishing the Good and the Bad",* Proceedings 2001 Winter Simulation Conference, IEEE Press, Dec. 2001, 95-105.

[39] Lewis T.G. y Payne W.H., *Generalized Feedback Register Pseudo-random Number Algorithm,* Journal of the Association for Computing Machinery, 1973.

[40] L 'Ecuyer P., Compagner A. y Cordeau J-F., *Entropy Tests for Random Number Generator",* Abstract, Dépt. d 'Informatique et de Recherche Opérationnelle, Université de Montréal, Canadá, 1996.

[41] L 'Ecuyer P., *Uniform random number generation,* Département d 'IRO, Université de Montréal, Annals of Operations Research 53, 1994.

[42] Marin I. y Palma R. J. A., *Manual básico de métodos de camino crítico,* Serie N ° 16 , Consejo Federal de Inversiones, Buenos Aires, 1973.

[43] NDP Fortran 386/DOS User's Manual, Microway Inc., Kingston, USA, 1993.

[44] Mathews Ilane L. y Karian Zaven, *The Generation and Testing of Pseudo-random Numbers on Computers*, Honors Project, Dept. of Mathematical Sciences, Denison University, 1986.

[45] Mendenhall W., Scheaffer R.L. y Wackerly, *Estadística matemática con aplicaciones*, Grupo Editorial Iberoamérica, México,1986.

[46] Microsoft MS-DOS version 6.0, *Manual del usuario*, Microsoft Corporation, USA, 1993.

[47] Minuteman Software, *GPSS/PC Reference Manual, General Purpose Simulation*, Minuteman Software, Stow, MA, 1986.

[48] Monagan M.B. et al., *MAPLE V Programming Guide, Waterloo Maple Inc.*, 1996.

[49] Naylor Thomas, *Experimentos de Simulación en Computadoras con Modelos de Sistemas Económicos*, Limusa, México, 1977.

[50] Naylor Thomas H. et al., *Computer Simulation Techniques*, New York, John Wiley & Sons, Inc., 1966.

[51] NDP Fortran 386/DOS, *User's Manual*, Microway Inc., Kingston, MA, USA, 1993.

[52] NDP Tools, *NDP 386/486Compilers*, Microway Inc., Kingston, USA, 1993.

[53] NDP Fortran Library, *Library Reference Manual and Languaje Reference Manual*, Microway Inc., Kingston, USA, 1993.

[54] Niederreiter Harald, *Random Number Generation and Quasi-Monte Carlo Methods*, Society for Industrial and Applied Mathematics, 1992.

[55] NDP Fortran Microway, *NDP Fortran Language Manual*, Microway Inc., 1993.

[56] Olmos Angel A. y Boché Silvia, *Estudio comparativo de generadores para distribuciones en redes de actividades*, I ELIO, OPTIMA'97, Univ. de Concepción, Chile, 1997.

[57] Olmos A. y Nellar R., *Programación y decisiones por el método de camino crítico*, Cuadernos de Economía y Administración, UNC, Serie N°3.

[58] Olmos A. y Arenas L., *Estudio comparativo de modelos generadores de cantidad de tormentas de granizo por meses*, serie Cuadernos de la F.E.A., U.N.C., N ° 12, 1989.

[59] Olmos A. A. y Allan M. C., *Generación de números aleatorios y tests estadísticos en microcomputadores - Parte 1*, Cuadernos de Investigación, Avances y Resultados Parciales, Serie Informática y Estadística, N°3, 1996.

[60] Olmos A., Arenas L., Allan C., *Estudio de la eficiencia de distintas aeronaves para combatir incendios forestales mediante simulación*, Cuarto Encuentro Académico Tecnológico, Resistencia (Chaco), 1993

[61] Payne W. H. et al., *Coding the Lehmer Pseudo-random Number Generator*, Communications ACM, 1969.

[62] Perez López C., *Análisis Estadístico con STATGRAPHICS*, Ra-Ma, Madrid, 1995.

[63] Ríos Insúa D. et al., *Simulación. Métodos y aplicaciones*, Ra-Ma, Madrid, 1997.

[64] Rico Rico, F., *La Aviación como medio de ayuda frente al incendio forestal*, Revista Aeronáutica y Astronáutica, Madrid, 1988.

[65] Schmeiser B., *Batch Size Effects in the Analysis of Simulation Output*, Operations Research, 30(1982), 556-568

[66] Schonberger R., *Why projects are "always" late: a rationale based on manual simulation of a PERT/CPM network*, Interfaces vol. 11, N°5, 1981.

[67] Schrage Linus, *A More Portable Fortran Random Number Generator*, ACM Transactions on Mathematical Software, 1979.

[68] Schriber Thomas J., *An Introduction to Simulation Using GPSS/H*, J.-Wiley, N.York, 1991.

[69] Strohmeir A., *FORTRAN 77*, Editores Técnicos Asociados, Barcelona, 1985.

[70] Swain C. Gardner y Swain Marguerite S., *A Uniform Number Generator That Is Reproducible*, Hardware-Independent and Fast, Journal of Chemical Information and Computer Sciences, 1980.

[71] Swain J. J., *Simulation Reloadead*, pp.46-49,ORMS Today, v.30-4, 2003

[72] Sylvester G. y Sosa Escalada J., *Pert y Gert*, Ed. autor, 1975.

[73] Taha Hamdy A., *Simulation Modeling and SIMNET,* Prentice Hall-International Ed., N.Jersey, 1988.

[74] Thesen Arne, *An Efficient Generator of Uniformly Distributed Random Variates Between Zero and One*, Simulation, 1985.

[75] Thomas G., *Introduction de l 'aléatoire dans les problemes d'ordonnancement. Méthode de simulation,* Monographies de Recherche Opéra - tionnelle, Dunod, Paris, 1968.

[76] Ventura E., *L'introduction de l'aleatoire dans les reseaux Pert*, Revue–Française de Recherche Opérationnelle, N°38, 1966.

[77] Whichman B.A. y Hill I.D., *Algorithm AS183:An efficient and portable pseudo-random number generator,* Applied Statistics, 31,188-190, 1982.

[78] Whittlesey John R. B., *A Comparison of the Correlational Behavior of Random Number Generators for the IBM 360,* Communications of the ACM, 1968.